Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Řídicí software pro zařízení měřící tlak

Control software for pressure measuring equipment

2020 **Bc. Marián Antoszyk**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Marián Antoszyk**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2612T041 Řídicí a informační systémy

Téma: Řídicí software pro zařízení měřicí tlak
Control Software for Pressure Measuring Equipment

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Analýza současného stavu HW řešení měřicího řetězce.
2. Návrh komunikačního protokolu mezi zařízením gateway a IR node.
3. Návrh komunikačního protokolu mezi zařízením gateway a centrální jednotkou.
4. Vytvoření firmware pro zařízení IR node.
5. Vytvoření firmware pro zařízení gateway.
6. Vytvoření řídicí aplikace pro centrální jednotku.
7. Implementace do celkového měřicího řetězce.
8. Testování a zhodnocení řešení.

Seznam doporučené odborné literatury:

[1] ZHU, Yifeng. *Embedded systems with arm cortex-m microcontrollers in assembly language and c.* 3rd edition. Ballston Spa, NY: E-Man Press, 2017. ISBN 978-0982692660.
[2] KRAUSE, Jörg. *Introducing Bootstrap 4.* New York, NY: Springer Science+Business Media, 2016. ISBN 978-1484223819.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jaromír Konečný, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky ID16 OSTRAVA

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *11. května 2020*

……
podpis studenta

Poděkování

Rád bych poděkoval svému vedoucímu práce, panu Ing. Jaromírovi Konečnému, Ph.D., za vedení, odbornou pomoc a konzultaci při vytváření této diplomové práce, zároveň také za možnost účastnit se několika projektů v rámci Katedry kybernetiky a biomedicínského inženýrství.

Abstrakt

Diplomová práce se zabývá realizací softwarové části pro měřicí systém tlaku. Měřicí systém je složen ze sady bezdrátových měřicích kolíků, sítě přijímačů, brány a řídicí jednotky. V teoretické části diplomové práce je popsána navržená struktura měřicího systému, hardwarová koncepce řešení, použité technologie pro komunikaci mezi zařízeními a základní teorie o softwarovém inženýrství. V praktické části je rozebrán návrh komunikačních protokolů, návrh a implementace firmwarů pro mikrokontrolery, realizace řídicí aplikace pro řídicí jednotku s operačním systémem linuxové distribuce, a nakonec také testování celého systému.

Klíčová slova

Měřicí systém, měření tlaku, software, firmware, programování, sběrnice, CAN, SPI, IR, komunikační protokol, mikrokontroler, MCU, C, C++, UML, Raspberry Pi

Abstract

The diploma thesis deals with the implementation of the software part for the pressure measurement system. The measurement system consists of a set of wireless measuring pins, a network of receivers, a gateway and a control unit. The theoretical part of the thesis describes the structure of the measurement system, hardware concept of the solution, technology for communication between devices and their standards. The practical part describes the process of designing and implementing firmware for microcontrollers, implementing a control application for a control unit with a Linux distribution operating system, and finally testing the entire system.

Key words

Measurement system, pressure measurement, software, firmware, programming, CAN bus, SPI, IR, communication protocol, microcontroller, MCU, C, C++, UML, Raspberry Pi

Obsah

Seznam použitých symbolů a zkratek

Zkratka/Symbol	Význam
AD	Analogově-digitální
API	Application Programming Interface – rozhraní pro programování aplikací
ARM	Označení architektury procesorů
ASK	Amplitude-shift keying – Klíčování amplitudovým posunem (forma modulace)
C	Programovací jazyk nízké úrovně
C++	Programovací jazyk
CAN	Controller Area Network – komunikační sběrnice využívaná v průmyslu
CRC	Cyclic Redundancy Check – cyklický redundantní součet (funkce k detekci chyb)
CSI	Camera Serial Interface – sériový port pro připojení kamery
DC	Direct current – stejnosměrný proud
DFD	Data-flow Diagram – diagram datových toků (nástroj pro modelování toku dat)
DPS	Deska plošných spojů
DSI	Display Serial Interface – rozhraní pro připojení displeje
ECU	Electronic Control Unit – vestavěný počítač pro řízení automobilových systémů
EEPROM	Electrically Erasable Programmable Read-Only Memory – Elektronicky vymazatelná paměť pouze pro čtení
EID	Extended Identifier – identifikátor zpráv u sběrnice CAN 2.0B
ER	Entity Relationship – vztah mezi entitami v softwaru
FSK	Frequency-shift keying – Klíčování frekvenčním posunem (forma modulace)
GPIO	General-Purpose Input/Output – univerzální vstupní/výstupní elektrický kontakt
GW	Gateway – zařízení v měřicím řetězci
HDMI	High-Definiton Multimedia Interface – zvukové a obrazové rozhraní
ID	Identifier – identifikátor
IEC	International Electrotechnical Commission – mezinárodní elektrotechnická komise
IR	Infrared Radiation – infračervené záření
ISO	International Organization for Standardization – mezinárodní organizace pro normalizaci
LED	Light-Emitting Diode – elektroluminiscenční dioda
MCU	Microcontroller Unit – mikrokontroler
NRZ	Non Return To Zero – kódování binárního signálu
PC	Personal Computer – osobní počítač

POSIX	Portable Operating System Interface – označení standardu používaného unixovými operačními systémy
RAM	Random Access Memory – polovodičová paměť s náhodným přístupem
SD	Secure Digital – paměťová karta používaná v přenosných zařízeních
SPI	Serial Peripheral Interface – sériové periferní rozhraní
SQL	Structured Query Language – standardizovaný strukturovaný dotazovací jazyk pro práci s daty v databázi
UART	Universal Asynchronous Receiver-Transceiver – Asynchronní sériová sběrnice
UML	Unified Modeling Language – standardizovaný modelovací jazyk pro software
USB	Universal Serial Bus – Univerzální sériová sběrnice

Seznam obrázků

Seznam tabulek

Úvod

Se stále rostoucím trendem digitalizace, novými technickými oblastmi těšícími se velkému nadšení a popularitě, jako je internet věcí, 3D tisk, cloudy, umělá inteligence a další, pronikají nové technologie do více a více oborů lidské činnosti. Souhrnně lze tyto oblasti technického pokroku označit (zejména marketingovým) pojmem průmysl 4.0, jehož hlavní doménou mají být kyberneticko-fyzikální systémy. Jedná se o výpočetní systémy schopné interagovat s fyzikální světem, jinak řečeno systémy schopné řídit určený technologický úsek, autonomně se rozhodovat a plnohodnotně zastoupit některý z výrobních či jiných procesů. Díky neustálému vývoji elektrotechnických a informačních technologií, které dávají vznik novým postupům a prostředkům, se naskýtá obrovský prostor pro vytváření inovativních řešení v jiných technologických odvětvích.

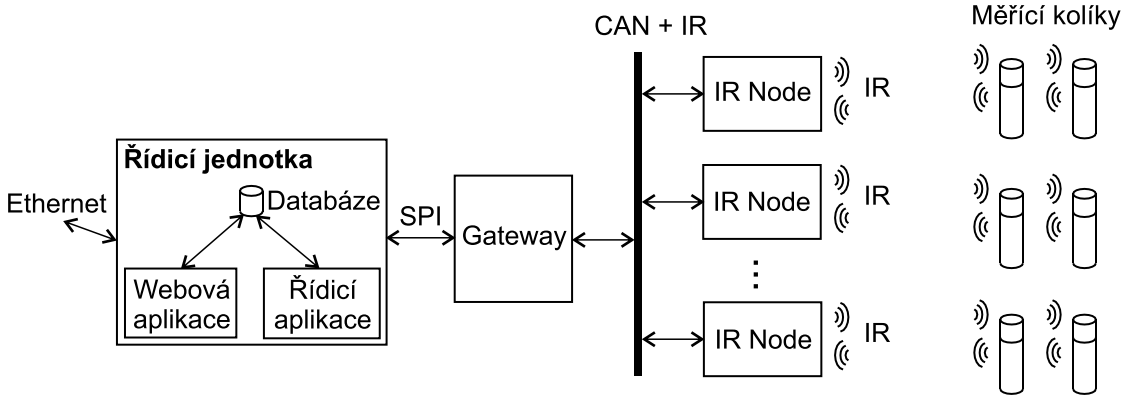
Téma této práce je vytvořeno na základě vědecko-výzkumného projektu, jehož cílem je vytvoření prototypu měřicího kolíku s integrovaným měřením upínací síly a bezdrátovým přenosem dat. Cílovou oblastí pro vyvíjený měřicí kolík je průmyslové prostředí, kde by se měl využívat pro měření působící síly v lisu pro plošné tváření. Důvodem pro řešení této problematiky je fakt, že vlivem špatně použité síly či při nerovnoměrném rozložení síly při lisování dochází k vadám na výsledném výrobku. Tím vznikají materiálové, finanční a časové škody.

Přínos diplomové práce je zejména ve vytvoření softwarové části měřicího řetězce a otestování celého systému. V práci není zahrnut návrh softwaru samotného měřicího kolíku, neboť projekt byl koncipován na více let a při realizaci diplomové práce byl již prototyp měřicího kolíku vytvořen. Nicméně měřicí kolík je jen jedna část celého měřicího systému, který se skládá z více různých zařízení, která budou v této práci představena.

V práci je nejprve představena koncepce celého měřicího systému, následně je věnována každému zařízení v měřicím řetězci jedna podkapitola. Tato zařízení jsou popsána z hlediska jejich funkcionality v měřicím systému a z hlediska hardwarového pohledu. V následujících dvou kapitolách jsou zpracovány teoretické informace týkající se prostředků souvisejících s praktickou částí. Nejprve je uveden obecný popis o použitých způsobech komunikace mezi jednotlivými zařízeními, po něm následuje kapitola pojednávající o softwarovém inženýrství a s tím souvisejících postupech při vytváření softwaru. Aby bylo umožněno mezi zařízeními komunikovat, je zapotřebí vytvořit komunikační protokoly pro jednotlivá komunikační média. Kompletní postup při návrhu komunikačních protokolů je popsán ve čtvrté kapitole. Následující 3 kapitoly se zabývají návrhem a implementací softwaru pro jednotlivá zařízení v měřicím systému. U každého zařízení jsou popsány použité nástroje pro vytváření softwaru, dále je uveden výčet funkčních požadavků, jejich následná analýza a návaznost na implementaci. Také jsou rozebrány některé implementační detaily, které byly při tvorbě softwaru zváženy. Poslední kapitola je věnována vytvoření testovacího měřicího systému z finálních prototypů jednotlivých zařízení a následně je popsáno provedení několika testů. V závěru jsou zhodnoceny dosažené výsledky v rámci této práce.

1 Analýza HW řešení měřicího řetězce

Výchozím bodem pro tvorbu této práce je analýza již realizovaného řešení. To bylo vytvořeno v rámci projektu, kterého je tato diplomová práce součástí. Před započetím práce byla již navržena koncepce měřicího systému (Obr. 1), na základě rešerše zvolen fyzikální princip měření působící síly, navržena koncepce mechanické konstrukce jednotlivých zařízení, proveden design elektroniky a také vytvořen návrh struktury uživatelského rozhraní. Zároveň již byly k dispozici první hardwarové prototypy elektroniky v počtu pár jednotek kusů.



Obr. 1 - Blokové schéma struktury měřicího systému

Jak je z výše uvedeného blokového schématu patrné, v měřicím řetězci se nachází 4 typy zařízení. Každé z těchto zařízení je detailněji popsáno v podkapitolách této kapitoly, následující text pojednává o zařazení těchto zařízení do obecné teorie měření a měřicích systému.

Na pravé straně schématu jsou zobrazeny **měřicí kolíky**, které v měřicím systému zaujímají roli senzorické části. Zároveň provádějí konverzi měřeného signálu z analogové podoby na digitální. Měřicí kolíky komunikují bezdrátově, a to za pomocí infračerveného záření.

Pro komunikaci s měřicími kolíky jsou v měřicím systému umístěny tzv. **IR nody**. Jedná se o zařízení, jenž umožňují fyzicky přijímat a vysílat data skrze infračervený přenos. Všechny IR nody jsou připojeny na sběrnici CAN.

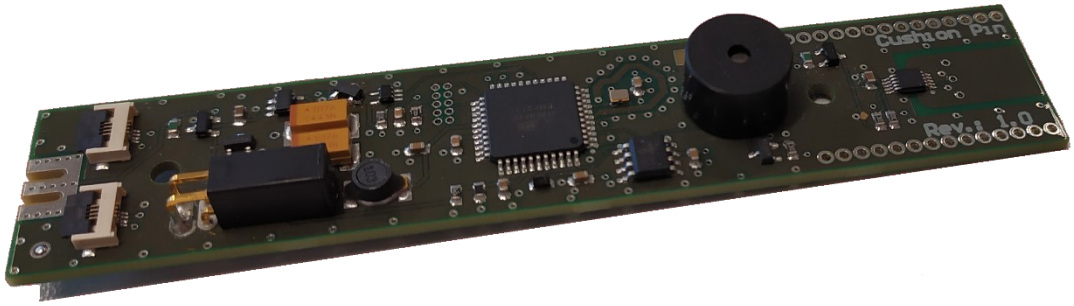
Na sběrnici CAN je také připojeno také další zařízení – **gateway**. Jak je z názvu patrné, jde o bránu, pomocí které je umožněno komunikovat se zařízeními v měřicím řetězci. IR nody spolu se zařízením gateway tvoří elementy měřicího systému sloužící pro přenos dat.

K zařízení gateway je přes sběrnici SPI připojeno poslední zařízení měřicího systému – **řídicí jednotka**. Jejím primárním úkolem je řízení měřicího systému, zpracování a uchovávání dat. Jelikož je uživatelské rozhraní měřicího systému v konceptu navrženo jako webová aplikace, která běží přímo v řídicí jednotce, slouží zároveň jako jednotka pro prezentaci dat.

1.1 Měřicí kolík

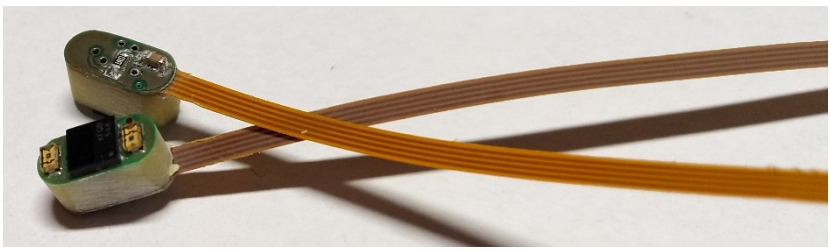
Měřicí kolík je zařízení v měřicím systému, které slouží pro měření působící síly pod lisem. V rámci zadání projektu bylo dohodnuto, že měřicí kolík bude mít válcový tvar s průměrem 30 mm a délkou 300 mm a bude umožňovat měření pro rozsah zatížení 0–80 kN. Zároveň byl kladen velký důraz na jeho mechanickou odolnost, jelikož je mechanicky namáhán lisem a je určen do průmyslového prostředí, kde nelze očekávat citlivé zacházení.

Každý měřicí kolík obsahuje vlastní elektroniku, která je do něj vložena. Velikost elektroniky je značně omezena mechanickou koncepcí měřicí kolíku, zejména pak jeho vnitřním průměrem, který je 24 mm. Měřicí kolík je napájen z vlastní baterie, tudíž byl při vývoji elektroniky brán ohled na jeho spotřebu. Příkladem je využití senzoru náklonu, který je na elektronice umístěn. K aktivaci elektroniky dojde při fyzickém překlopení měřicího kolíku do aktivní polohy (vertikální poloha). Samozřejmostí je, že elektronika obsahuje také mikrokontroler, kterým je měřicí kolík řízen.



Obr. 2 - Prototyp elektroniky měřicího kolíku

Měřicí kolík komunikuje s okolím bezdrátově za využití IR přenosu. Aby bylo měřicímu kolíku umožněno komunikovat, jsou ve stěnách mechanické konstrukce vytvořeny otvory, ve kterých jsou umístěny IR vysílače a přijímače. Ty nejsou součástí výše uvedené elektroniky, ale jsou vyrobeny metodou flex-rigid DPS a k elektronice jsou připojeny. Princip komunikace pomocí infračerveného záření je popsán dále v kapitole 2.1. Mimo IR komunikaci měřicí kolík využívá také pomocnou zvukovou signalizaci.



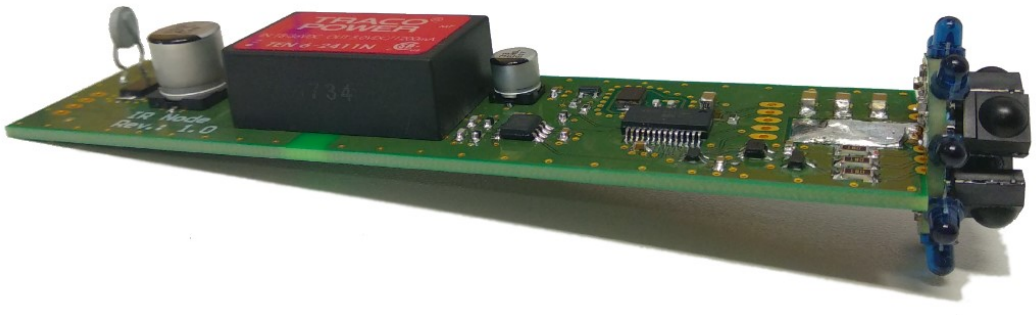
Obr. 3 - IR vysílače a přijímače pro měřicí kolík

1.2 IR node

Ve struktuře celého měřicího systému IR node zajišťuje předání dat z IR přenosu (data zasílaná z měřicích kolíků) na sběrnici CAN a fyzické vysílání IR příkazů směrem k měřicím kolíkům. Jak je patrné z anglického významu slova node, jedná se v podstatě o přenosový uzel mezi měřicími kolíky a centrální jednotkou. V měřicím systému se počítá s využitím několika IR nodů, které budou rozmístěny různě v prostoru okolo měřicích kolíků.

Z technického hlediska je vhodné, aby pokrytí infračerveného záření při vysílání dat bylo co možná největší, tedy aby byla co největší pravděpodobnost úspěšného přenosu dat k vybranému měřicímu kolíku v měřicím systému. Proto bylo navrženo, aby IR nody vysílaly data k měřicím kolíkům synchronně. Toho není možné dosáhnout způsobem, kde by byla IR nodům předána data hromadnou zprávou a ty by začaly tato data okamžitě vysílat, neboť i nepatrné časové zpoždění mezi IR nody při vysílání způsobí kolizi a tím neúspěšné zaslání dat. Vysílání dat pomocí IR je tedy zajištěno fyzickou vrstvou CAN sběrnice. V době vysílání dat přes IR je potřeba IR node odpojit od sběrnice (nikoliv fyzicky, ale softwarově) a umožnit přenos IR dat. Detailní popis tohoto principu je uveden v následujících kapitolách této práce.

Každý IR node je vybaven vlastní elektronikou. Napájení elektroniky IR nodu je přivedeno z externího zdroje, tudíž není důležité brát přílišný ohled na jeho spotřebu. Elektronika IR nodu je taktéž vybavena vlastním mikrokontrolerem. Pro komunikaci s měřicími kolíky za pomocí infračerveného signálu je IR node vybaven IR vysílači a přijímači, které jsou rozmístěny tak, aby pokrývaly úhel 360°. Vysílací a přijímací prvky jsou umístěny na separátní DPS a vytvářejí „terčík“, který je k DPS s ostatní elektronikou připájen pod úhlem 90°. Na níže uvedeném obrázku je fotografie prototypu první verze elektroniky, která byla při řešení této práce k dispozici. Návrh elektroniky a dostupný prototyp byl vytvořen v rámci bakalářské práce Bc. Danielem Kajzarem. [1]



Obr. 4 - Prototyp elektroniky IR nodu [1]

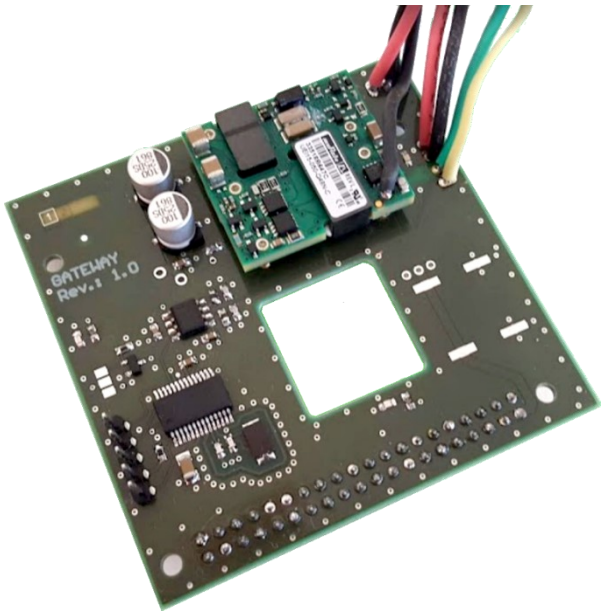
1.3 Gateway

Zařízení s názvem gateway zajišťuje komunikaci mezi řídicí jednotkou a IR nody či měřicími kolíky. Také zde lze z anglického významu slova gateway odvodit, že je jedná doslova o bránu mezi řídicí jednotkou a ostatními zařízeními v měřicím systému.

Z technického pohledu zařízení gateway zprostředkovává řídicí jednotce komunikaci na sběrnici CAN, kterou multiplexuje s vysíláním dat skrze IR přenos. V momentě vysílání dat IR přenosem musí být všechna ostatní zařízení od sběrnice CAN softwarově odpojena, aby nedocházelo k poškození vysílaných dat. Díky využití tohoto způsobu přenosu IR dat je odstraněno jakékoli zpoždění, ke kterému by mohlo dojít při řízení IR vysílání jednotlivými IR nody. Tím se výrazně omezuje prostor pro možné chyby v komunikaci.

Komunikace mezi zařízením gateway a řídicí jednotkou je zajištěna pomocí sériového rozhraní SPI. Tato komunikace zajišťuje dostatečnou datovou propustnost, neboť přenosová frekvence dat se může pohybovat v řádech MHz.

Elektronika zařízení gateway je konstruována s ohledem na výběr řídicí jednotky. Hlavním bodem návrhu byl požadavek, aby bylo možno zařízení gateway spojit s řídicí jednotkou pomocí GPIO pinů a obě zařízení tak fyzicky tvořila ucelený celek. Napájení je k zařízení gateway přivedeno z externího zdroje, tudíž opět není potřeba věnovat důraz jeho spotřebě. Také i zde elektronika zahrnuje mikrokontroler. Na níže uvedené fotografii (Obr. 5) je zachycen prototyp elektroniky, který byl v době vytváření této práce k dispozici.

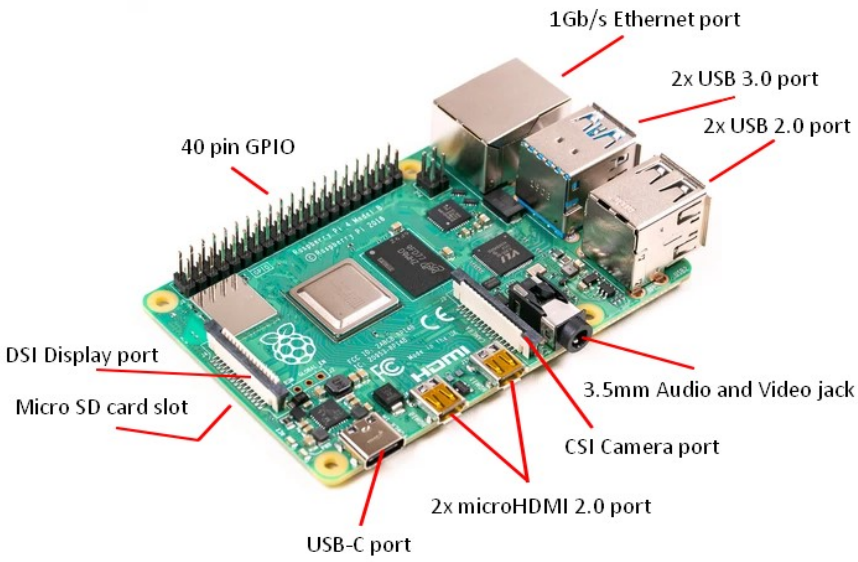


Obr. 5 - Prototyp elektroniky zařízení gateway

1.4 Řídicí jednotka

Řídicí jednotka (tj. centrální jednotka) slouží pro řízení měřicího systému. Byť fyzicky komunikuje pouze se zařízením gateway přes sériové rozhraní SPI, rozhoduje o tom, s kterým zařízením v měřicím řetězci bude provedena komunikace, monitoruje aktivní zařízení v měřicím systému, ale také umožňuje komunikaci s uživatelem, uchovává naměřená data atd. Kompletní výčet funkcionalit je uveden v kapitole 7.1.

Hardwarem řídicí jednotky celého měřicího řetězce je mikropočítač Raspberry Pi. Jedná se o malý jednodeskový počítač dostupný v několika variantách, které se odvíjí zejména podle historického vývoje zařízení. Současně nejnovějším produktem jsou modely čtvrté generace, která byla na trh uvedena v červnu 2019. Nová generace je vybavena 64bitovým čtyřjádrovým procesorem ARM Cortex-A72 s taktem 1,5 GHz a modelech dle velikosti paměti RAM - 1 GB, 2 GB nebo 4 GB. Zařízení je napájeno z externího zdroje nově přes USB-C port. Nově došlo také k nahrazení HDMI portu dvěma micro-HDMI porty a dvou ze čtyř USB 2.0 portů za dva USB 3.0 porty. [2][3]



Obr. 6 - Raspberry Pi 4 model B

Platforma Raspberry Pi podporuje širokou škálu operačních systémů, přičemž většinu lze pořídit bezplatně. Samotnou nadací oficiálně vyvíjený operační systém distribuce Linux se jmenuje Raspbian. Ten je odvozen z operačního systému Debian, který je vyvíjen dobrovolníky a nadšenci z celého světa. Raspberry Pi nedisponuje interním uložištěm, proto je nutné operační systém nahrát na externí uložiště (SD kartu, USB flash disk…) a připojit jej k zařízení. [3]

Konkrétním modelem použitým v řešení pro řídicí jednotku je Raspberry Pi 3 model B+. Důvodem, proč nebyl použit model nejnovější řady je, že hardwarová koncepce řešení vznikala ještě před datem, kdy byla čtvrtá řada uvedena na trh. Jelikož je zařízení gateway připojeno k řidicí jednotce přes GPIO piny, jehož rozložení se mezi modely 3. a 4. řady nezměnilo, je umožněno na novější řadu Raspberry Pi bez problémů přejít.

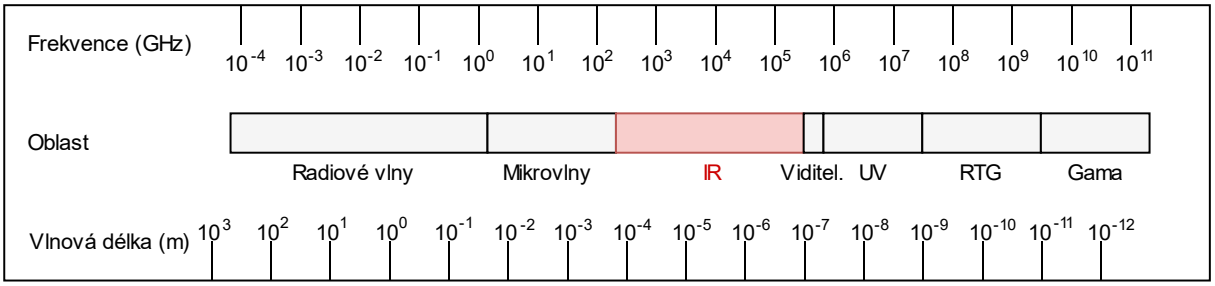
2 Principy přenosu dat mezi zařízeními

Přenosem dat mezi zařízeními se rozumí přenos digitálních zpráv nebo digitalizovaného analogového signálu. Pro přenos dat z jednoho zařízení na druhé je potřeba využít elektrické nebo elektromagnetické signály a také přenosové médium, což může být například metalický kabel, optický kabel, počítačové sběrnice, bezdrátový přenos atd.

Komunikace mezi zařízeními může být realizována jednosměrně nebo obousměrně. Jednosměrná komunikace odborně nazývaná jako simplexní je v praxi využívána například pro přenos dat z různých senzorů nebo například pro přenos dat do zobrazovacího zařízení. Pokud simplexní komunikace není dostačující a je potřeba mezi zařízeními provádět výměnu dat, je nutné volit komunikaci duplexní, jinými slovy obousměrnou. Ta může být plně duplexní, což označuje současnou obousměrnou komunikaci, nebo poloduplexní, což je obousměrná komunikace, u které v jednu chvíli jedno zařízení vysílá a druhé přijímá a naopak, avšak nikdy neprobíhá komunikace oběma směry zároveň. [4]

2.1 IR datový přenos

Infračervené záření je elektromagnetické záření s vlnovou délkou mezi 760 nm a 1 mm, respektive frekvencí mezi 300 GHz a 430 THz, což je mimo hranici spektra viditelného lidským okem. Na následujícím obrázku je zobrazeno umístění infračerveného záření v elektromagnetickém spektru.



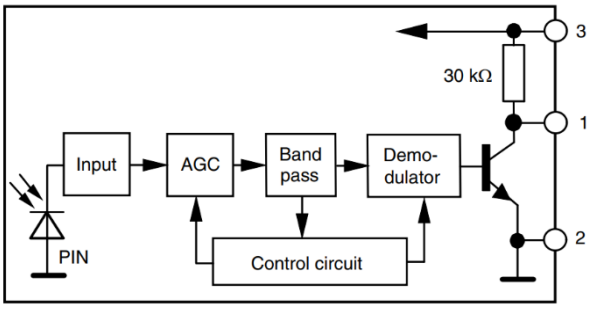
Obr. 7 - Elektromagnetické spektrum

Přenos dat mezi jednotkami pomocí infračerveného záření je jednou z nejlevnějších forem bezdrátového přenosu. Běžně se tento princip využívá ve spotřebitelské elektronice, například pro ovládání zařízení pomocí dálkových ovladačů. V dřívějších letech byl rozsah využití IR komunikace větší, postupně jej však v mnoha zařízeních nahradily jiné technologie jako například Bluetooth. [5]

Pro přenos dat pomocí infračerveného záření je nutné mít zařízení vybavená IR vysílači a IR přijímači. Tyto obvody se starají o převod elektrického signálu na optické záření v IR spektru a naopak. Infračervené záření je vysíláno pouze v určitém úhlu, z toho důvodu je důležité, aby byl vysílací a přijímací blok k sobě orientován. [5]

Vysílací blok tvoří IR LED, což je dioda emitující optické záření v IR spektru. Pro možnost přenosu na co největší vzdálenost je požadováno, aby měla IR LED co největší výkon. Ten je však omezen jednak maximálním proudem, který může IR LED protékat, a dále také faktem, že IR přenos využívají často zařízení, která jsou napájena z baterie či akumulátoru. [5]

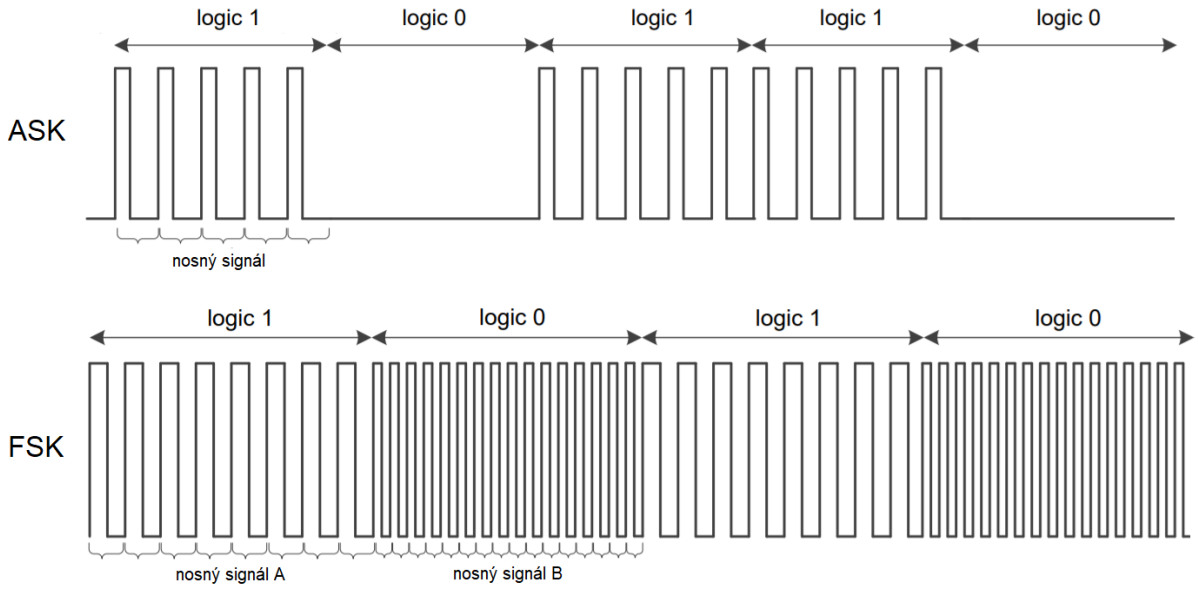
Přijímací blok se skládá z několika částí v závislosti na typu přijímače. Tyto části jsou obvykle integrovány v jednom pouzdře. Základ tvoří přijímací IR fotodioda. Ta však registruje širší pásmo IR spektra, někdy i část viditelného spektra. Jelikož signál mimo vlnovou délku vysílače lze označit jako rušení, musí být přijímací část přizpůsobena. Z tohoto důvodu zahrnuje přijímací blok také pásmový filtr, který propustí signál pouze určité vlnové délky. Na následujícím obrázku je uvedeno blokové schéma elektroniky konkrétního IR přijímače dostupného na trhu. [5]



Obr. 8 - Blokové schéma IR přijímače TSOP32238 [6]

Signál vysílaný vysílačem pomocí IR LED bývá modulován. Modulace pomáhá zejména přijímací části, která dokáže lépe oddělit užitečný signál od okolního rušení. Princip modulace spočívá ve změně nosného signálu, což bývá sinusový, obdélníkový či jiný periodický signál pomocí modulačního signálu, což je signál nesoucí informaci (data). [7]

V praxi se pro IR přenos používá nejčastěji FSK modulace či ASK modulace. Modulačním signálem pro IR přenos je unipolární NRZ kódovaný datový signál. Nosným signálem je obdélníkový signál. Na následujícím obrázku (Obr. 9) jsou zobrazeny příklady modulovaných signálů využívaných pro přenos pomocí IR. Prvním typem je modulovaný signál pomocí ASK modulace metodou ON-OFF klíčování. Druhým typem je modulovaný signál pomocí FSK modulace. [7]



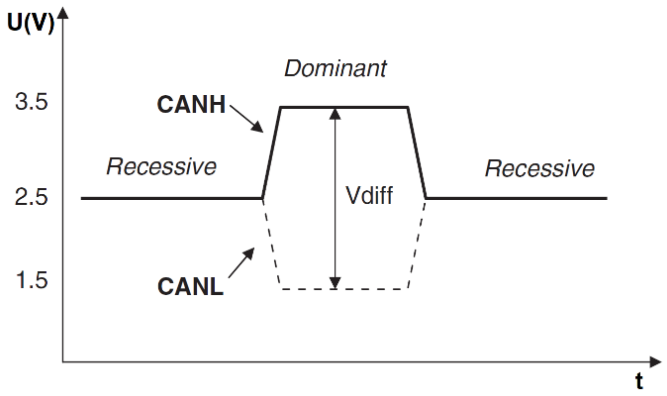
Obr. 9 - Příklady modulovaných signálů pro přenos IR dat [7]

2.2 Sběrnice CAN

Sběrnice CAN (Controller Area Network) je sériová datová sběrnice vyvinutá firmou Robert Bosch GmbH. Protokol sběrnice byl vydán v roce 1986, první čipy byly vyrobeny o rok později společnostmi Intel a Phillips. Elektrické parametry fyzického přenosu jsou definovány mezinárodními normami ISO 11898 a ISO 11519. V dnešní době se sběrnice CAN využívá v průmyslové automatizaci, lékařských přístrojích, automatizaci budov, výrobních strojích a mnoha dalších odvětvích, nicméně jeho první a zároveň nejčastější aplikační oblastí je automobilový průmysl. V této oblasti se využívá pro vnitřní komunikaci mezi senzory, aktuátory a řídicími jednotkami (ECU) v reálném čase. [8]

Nespornou výhodou sběrnice CAN je, že se jedná o sběrnici typu multi-master. Každá jednotka připojená na sběrnici je v roli master, tím pádem může začít kdykoliv vysílat, musí však být sběrnice volná (nesmí právě vysílat jiná jednotka). Řešením kolizí při souběžném vysílání více jednotek je věnována podkapitola 2.2.2. [9]

Zařízení se na sběrnici CAN připojuje pomocí čtyř vodičů, které jsou označeny jako Vcc, GND, CANH a CANL. Přenos dat probíhá diferenciálně, logická hodnota 0 je v terminologii CAN označena jako „dominant“, logická 1 jako „recessive“. Napěťové úrovně logických hodnot jsou určeny pomocí prahového napětí, kde při stavu „dominant“ je rozdíl napětí mezi CANH a CANL větší než prahové napětí, při stavu „recessive“ je tomu naopak. Tento princip je znázorněn na následujícím obrázku. [8]



Obr. 10 - Rozlišení logických úrovní na sběrnici CAN [8]

Přenosové rychlosti na sběrnici CAN se mohou pohybovat v závislosti na verzi protokolu do 1 Mbit/s u normální verze CAN, u verze CAN-FD („Flexible Datarate“) až do 5 Mbit/s. Podle volby přenosové rychlosti se také odvíjí maximální délka sběrnice. Například pro rychlost 1 Mbit/s se uvádí délka 40 metrů, pro poloviční rychlost, tedy 500 kbit/s, je uváděno 100 metrů. Pro zamezení odrazů signálu na sběrnici je potřeba provést impedanční přizpůsobení pomocí terminace. Norma ISO 11898 požaduje charakteristickou impedanci sběrnice 120 Ω. Standardním způsobem terminace je využití rezistorů hodnoty 120 Ω na obou koncích sběrnice, ale lze využít i terminace zvané „Split“ nebo „Biased Split“. [8][9]

2.2.1 Struktura CAN rámců

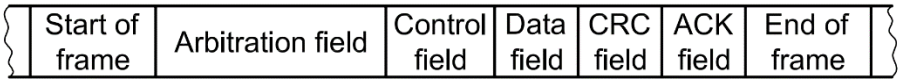
Dle standardní verze CAN se mohou na sběrnici CAN vyskytovat 4 typy rámců. V terminologii jsou tyto rámce označeny jako:

- Data frame - Rámec sloužící pro přenos dat mezi jednotkami.
- Remote frame - Slouží pro přijímací jednotku k vyžádání přenosu od vysílací jednotky.
- Error frame - Rámce generované CAN hardwarem indikující chybu během přenosu.
- Overload frame - Rámec generovaný přijímací jednotkou indikující, že není připravena přijímat.

CAN protokol obsahuje dva typy datových rámců označených jako 2.0A a 2.0B. CAN 2.0A je původní standard, kde délka identifikátoru datového rámce má 11 bitů, zatímco novější standard 2.0B má délku 29 bitů. Novější verze 2.0B je plně kompatibilní s verzí 2.0A, takže dokáže zpracovávat zprávy v obou standardech. [9]

Datový rámec obou verzí standardů se skládá z těchto částí:

- Start of frame - 1 bit určující začátek rámce.
- Arbitration field - Arbitrážní pole rámce též označované jako identifikátor rámce ID.
- Control field - Obsahuje 2 vyhrazené bity a 4 bity pro určení délky datového pole (Data field)
- Data field - Datové pole o velikosti 0 - 64 bitů.
- CRC field - Cyklický redundandní součet rámce sloužící pro kontrolu správnosti přenosu.
- ACK field - Pole o velikosti 2 bitů pro potvrzení přijetí rámce jiným zařízením/zařízeními.
- End of frame - 1 bit určující konec rámce.



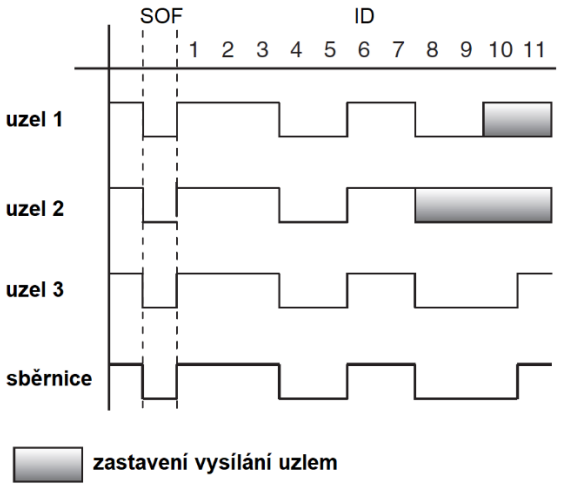
Obr. 11 - Standardní datový rámec [9]

Při přenosu rámce na sběrnici CAN je využívána technika zvaná „bit stuffing“. Tento pojem lze volně přeložit jako vkládání bitů. Jakmile je přeneseno 5 po sobě jdoucích bitů stejné logické úrovně, dojde k přenosu jednoho bitu opačné logické úrovně. Tato metoda slouží k re-synchronizaci všech jednotek na sběrnici. [8]

2.2.2 Arbitráž na sběrnici

Jak již bylo zmíněno dříve, sběrnice CAN je typu multi-master, a tedy je-li sběrnice volná, může každé zařízení začít na sběrnici vysílat data. Z toho je patrné, že může docházet ke kolizím, kdy ve stejný okamžik začne více než jedno zařízení vysílat na sběrnici data. Pro tento případ má CAN standard definován postup, jak situaci řešit na úrovni fyzické vrstvy, aniž by došlo k poškození přenášených dat. Hlavní roli při tom hraje identifikátor datového rámce (ID). Nastane-li souběžné vysílání dat více zařízeními, probíhá arbitráž v závislosti na hodnotě identifikátoru tak, že zařízení porovnávají své právě vysílané bity s úrovní na sběrnici. V případě, kdy jedno zařízení provádí vysílání stavu „dominant“ (log. 0) a druhé vysílá stav „recessive“ (log. 1), dojde k tomu, že zařízení vysílající stav „recessive“ přeruší vysílání a vyčká na uvolnění sběrnice. Z toho je patrné, že největší prioritu na sběrnici má

identifikátor rámce s hodnotou 0. Zároveň je však důležité, aby nenastala situace, kdy budou různá zařízení vysílat datový rámec se stejnou hodnotou identifikátoru. V takovém případě by k vyřešení kolize nedošlo a zbytek datového rámce by byl poškozen. Celý výše popsaný princip je znázorněn na následujícím obrázku. [8]

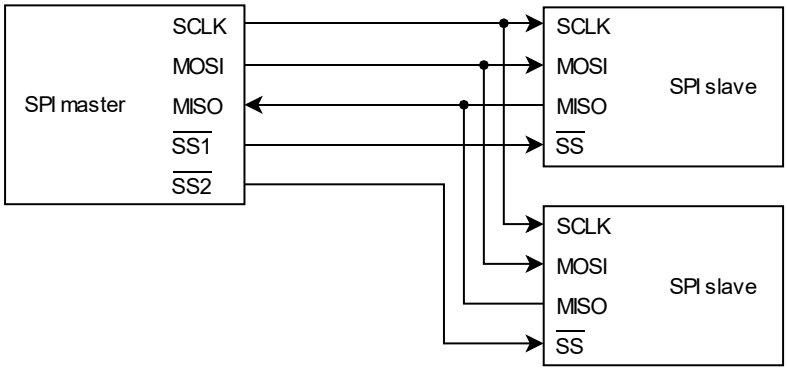


Obr. 12 - Princip arbitráže na sběrnici CAN [8]

2.3 Sběrnice SPI

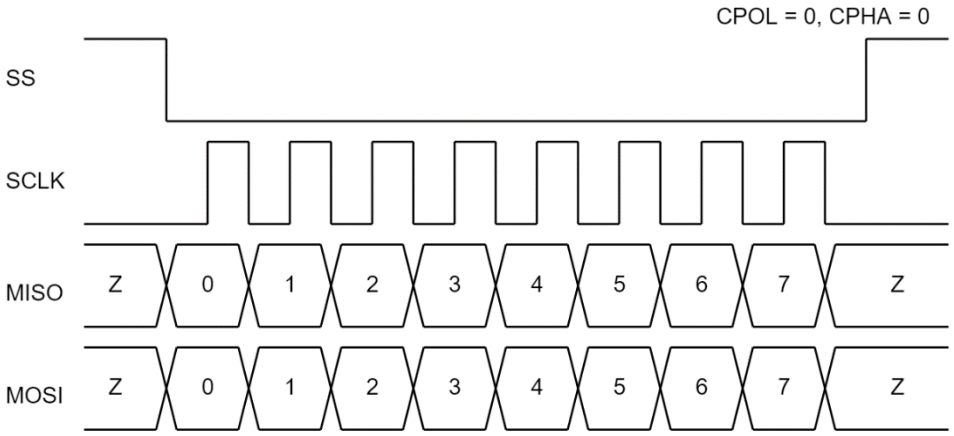
SPI označuje sériové komunikační rozhraní, které se využívá pro komunikaci mezi jednotkami na krátkou vzdálenost, typicky mezi obvody na desce plošného spoje. Typologie sběrnice vychází z modelu master-slave, kde jedna z jednotek zaujímá roli master, ostatní jednotky mají roli slave. Při standardním zapojení probíhá komunikace vždy mezi jednotkou master a jednou jednotkou slave v plně duplexním spojení. Výjimkou je zřetězené zapojení slave jednotek, kde komunikace probíhá jiným způsobem. Jednotky na sběrnici SPI se připojují pomocí 4 vodičů, kterými jsou:

- SCLK - „Serial Clock“, hodinový signál generovaný jednotkou v roli master.
- MOSI - „Master Output Slave Input“, datový výstup z jednotky v roli master.
- MISO - „Master Input Slave Output“, datový vstup do jednotky v roli master.
- SS - „Slave select“, signál pro výběr jednotky.



Obr. 13 - Standardní propojení 3 zařízení na sběrnici SPI

Začátek komunikace mezi jednotkami určuje jednotka v roli master. Ta aktivuje vybranou jednotku slave tak, že na vodiči SS změní logickou úroveň z hodnoty 1 na hodnotu 0 a zároveň začne generovat hodinové impulzy na vodič SCLK. S každým hodinovým impulzem je mezi jednotkami vyměněn 1 bit a to tak, že pomocí vodiče MOSI probíhá přenos bitu z jednotky master do jednotky slave, vodičem MISO je zaslán bit z jednotky slave do jednotky master. Přenos dat mezi jednotkami probíhá tak dlouho, dokud jednotka master nepřestane generovat hodinový signál nebo dokud na vodič SS nenastaví zpět logickou úroveň 1. [10]



Obr. 14 - Princip komunikace na sběrnici SPI

Jedním z důležitých detailů při komunikaci na sběrnici SPI je nastavení módu, který je udáván polaritou a fází hodinového signálu (SCLK). Tento mód se definuje v jednotkách standardně pomocí kombinace dvou konfiguračních bitů označovaných jako CPOL a CPHA. Polarita hodinového signálu (CPOL) udává, při které hraně (vzestupné nebo sestupné) budou jednotky data zapisovat. Fáze hodinového signálu (CPHA) udává logickou úroveň na vodičích MISO a MOSI při klidovém stavu. Pomocí těchto dvou parametrů lze vytvořit 4 kombinace, které jsou definovány jako módy SPI sběrnice. V následující tabulce je uvedeno označení těchto módů v závislosti na dané kombinaci parametrů hodinového signálu. [10]

Tabulka 1: SPI módy

SPI mód	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

3 Tvorba softwaru

Technickou disciplínou zabývající se všemi aspekty produkce softwaru je softwarové inženýrství. Tento obor vznikl na začátku 70. let 20. století. Cílem softwarového inženýrství je podpora profesionální tvorby softwaru. Mezi klíčové parametry, kterými se softwarové inženýrství zabývá, patří softwarový proces, spolehlivost, bezpečnost, požadavky na softwarový produkt a jeho opakované použití. [11]

Systematický přístup, který se uplatňuje při tvorbě softwarového produktu, se nazývá softwarový proces. Proces se skládá ze čtyř základních aktivit:

- Specifikace softwaru – definování funkce softwaru a omezení na jeho činnost.
- Vývoj softwaru – analýza, návrh a implementace softwaru.
- Validace softwaru – testování vyvinutého softwaru.
- Evaluace softwaru – vyhodnocení softwaru (zda odpovídá specifikaci).

Softwarový proces lze rozdělit na plánovaný proces a agilní proces. U plánovaného procesu jsou všechny aktivity předem naplánovány a postup při tvorbě se porovnává s plánem. U agilních procesů je plánování inkrementální a lze proces měnit a přizpůsobovat tak, aby byl schopen reagovat na změny požadavků. Každý z těchto přístupů se hodí pro odlišné typy softwaru v závislosti na technických, organizačních, projektových a dalších potřebách. [11]

Softwarový proces lze dále rozdělit také dle modelů. Tyto modely znázorňují softwarový proces z různých úhlů pohledu a vysvětlují různé přístupy k tvorbě softwaru. Mezi často používané modely softwarového procesu patří vodopádový model, inkrementální model, spirálový model a extrémní programování. Modely se vzájemně nevylučují a při vývoji velkých softwarových systémů se využívají jejich různé kombinace. [11]

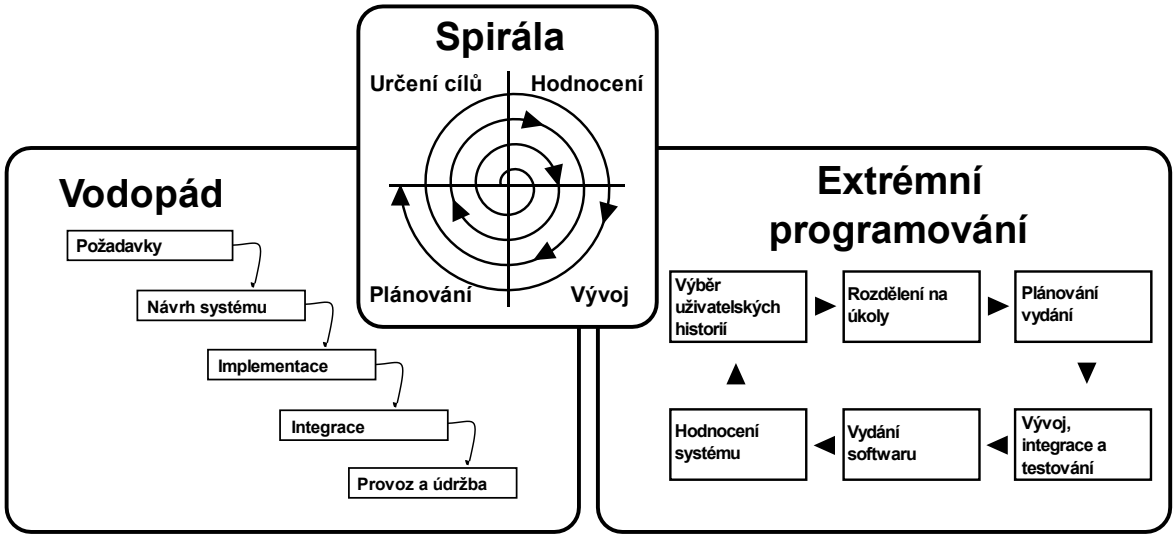
Prvním zmíněným přístupem je vodopádový model, jehož princip spočívá v tom, že množina činností dané fáze tvorby nemůže být zahájena dříve, než skončí fáze předešlá. Vlastností tohoto modelu je, že se výsledek vysoce odvíjí od specifikace systému a proces těžce reaguje na změny požadavků během tvorby softwarového díla. Výhodou modelu je, že klade velký důraz na dokumentaci a dokáže odhalit možné chyby již v rané fázi tvorby. Jako nevýhoda modelu je zmiňována nemožnost odhalení kvality softwaru dříve, než je software hotov, což je až v posledních fázích tohoto modelu. [11]

Druhým používaným modelem je inkrementální model, který je zaměřen na postupné vytváření verzí softwarového systému, u nichž přibývá počet funkcí, které mohou být definovány i v průběhu jeho vytváření. Dá se říct, že se jedná o iterace menších vodopádů s kratším trváním, kde každá iterace zahrnuje novou sadu doplňujících požadavků. Výhodou je možnost reagovat na změny požadavků během tvorby softwaru a předvést jeho funkcionalitu na konci každé iterace (s každou vydanou verzí). [11]

Spirálový model se zaměřuje na analýzu a minimalizaci rizik spojených s tvorbou softwarového systému. Stejně jako inkrementální model rozděluje projekt na menší segmenty opakující se iteračně. Model zahrnuje fáze plánování, analýzu cílů a rizik, vývoj a testování a vyhodnocení. [11]

Posledním zmíněným modelem je extrémní programování, což je agilní metoda tvorby softwaru. Hlavním cílem modelu je maximálně přizpůsobit proces změnám zákaznických požadavků,

a přitom produkovat kvalitní software. Mezi praktiky využívané v extrémním programování patří programování ve dvojicích, časté revize kódu, jednotkové testování kódu, programování pouze toho, co je zrovna nutné a jiné specifické činnosti. [11]



Obr. 15 – Grafické znázornění některých modelů softwarového procesu

3.1 Modelování softwarových systémů

Aby bylo umožněno zachytit odlišný pohled či perspektivu na vytvářený softwarový systém, používají se při jeho vývoji modely. Tyto modely jsou založeny zejména na grafickém znázornění systému. Modely slouží například pro odvození požadavků kladených na systém, během fáze návrhu slouží pro vývojáře, kteří software implementují, a v závěrečné fázi také pro dokumentaci struktury a fungování celého systému nebo jeho částí. Důležitým prvkem při modelování je dodržet abstraktní pohled na systém, tedy vynechat podrobnosti a zaměřit se pouze na nejdůležitější vlastnosti. [11]

3.1.1 UML

Pro modelování se nejčastěji využívá jazyk UML (Unified Modeling Language), který je uznáván jako standardní přístup při tvorbě softwarových systémů. V současné době je užíván standard UML 2, který byl dokončen v roce 2004. Pro vytváření modelů poskytuje jazyk UML různé diagramy, což je nejčastěji používaná část standardu. Diagramy jsou rozděleny do dvou kategorií na diagramy strukturální (též statické) a diagramy chování (též dynamické). Do zmíněných kategorií spadají konkrétní diagramy zachycující softwarový systém z různého úhlu pohledu. Mezi digramy jazyka UML patří:

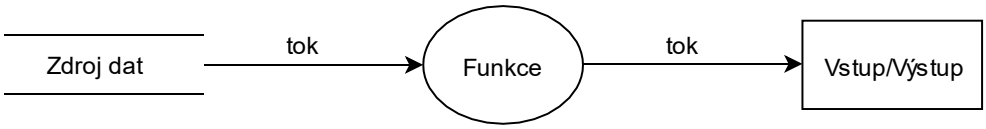
- **Strukturální diagramy** – diagram tříd, diagram komponent, objektový diagram, diagram rozmístění, diagram balíčků, diagram struktury.
- **Diagramy chování** – diagram případů užití, stavový diagram, diagram aktivit, sekvenční diagram, diagram spolupráce, diagram časování, komunikační diagram, diagram interakcí.

Podtržené diagramy ve výčtu jsou jedny z nejčastěji užívaných diagramů UML. Úroveň podrobnosti jednotlivých diagramů závisí na okolnosti použití. Běžně se tyto modely používají jako základ

pro diskuzi o systému, pro dokumentaci vytvořeného systému nebo jako podrobný popis, pomocí něhož je vytvářena implementace systému. [12][13]

3.1.2 **DFD**

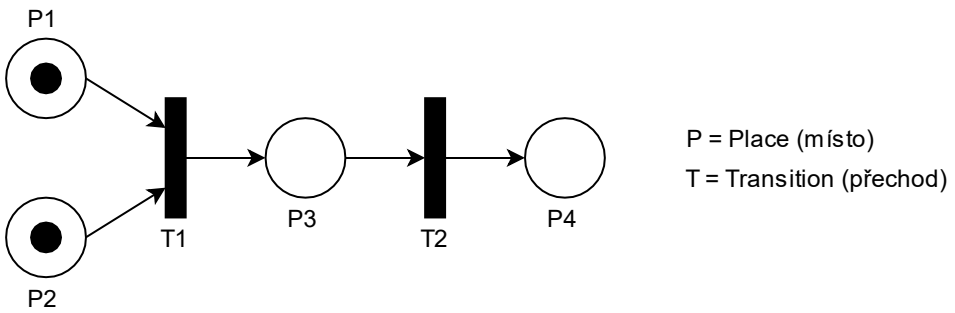
Další možností pro modelování softwarových systémů je využití DFD (Data-flow diagram) neboli diagramu datových toků. Model reprezentuje tok dat v procesu nebo systému a poskytuje informaci o vstupech a výstupech jednotlivých entit. Při modelování se využívá 4 základních prvků, kterými jsou proces (funkce), vstup/výstup (terminátor), zdroj dat (též sklad) a tok dat. Na následujícím obrázku je uvedena vizuální interpretace těchto prvků v notaci Yourdon. Notací pro modelování pomocí DFD existuje více, obvykle se mezi sebou liší jen ve vizuální reprezentaci prvků. [14]



Obr. 16 - Prvky DFD

3.1.3 **Petriho sítě**

Petriho sítě jsou nástrojem pro modelování paralelních a distribuovaných systémů. Vznikly v roce 1962 a jsou definovány mezinárodní normou ISO/IEC 15909. Jejich princip je matematicky založen, model lze použít k jednoznačné specifikaci dané aplikace nezávisle na volbě implementačních prostředků (hardwaru a softwaru). Petriho síť je orientovaný biparitní graf využívají grafické prvky jako jsou uzly, což může být místo (place) nebo přechod (transition), a orientované hrany. Orientované hrany v Petriho síti propojují místa a přechody, nikdy nejsou orientovanou hranou propojeny dvě místa nebo dva přechody. Dalším prvkem je token, což je značka, kterou lze umístit do vstupního místa (place) a lze ho tzv. odpálit neboli přemístit do místa výstupního, jsou-li splněny podmínky přechodu. Na následujícím obrázku je uveden příklad modelu Petriho sítě, ve které se nachází 4 místa, 2 přechody, 5 orientovaných hran a 2 tokeny v místech P1 a P2. [15]

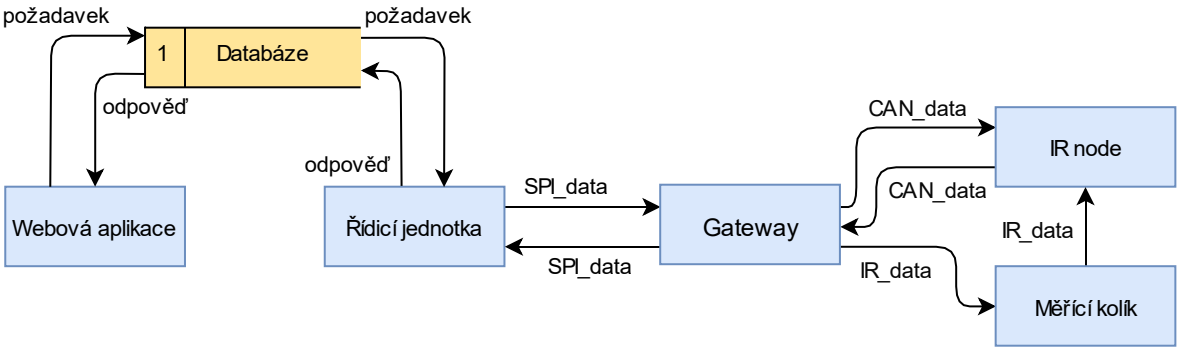


Obr. 17 - Příklad Petriho sítě

4 Návrh komunikace mezi zařízeními

Komunikace je v měřicím systému zavedena obousměrně. Řešení pomocí jednosměrné komunikace by bylo technicky obtížné realizovat, jelikož měřicí kolíky nemohou vysílat pomocí IR přenosu souběžně a bylo by potřeba řešit jejich synchronizaci, která by však vedla k menší či větší chybovosti v závislosti na počtu měřicích kolíků v systému nebo v závislosti na použitém synchronizačním algoritmu. Obousměrná komunikace tento problém dokáže eliminovat zavedením systému vyzývání, což znamená, že měřicí kolíky mohou komunikovat (přenášet data) pouze tehdy, jsou-li k tomu řídicí jednotkou vyzvány. Toto řešení však s sebou nese i nevýhodu v podobě větší režie při řízení komunikace na úrovni řídicí jednotky.

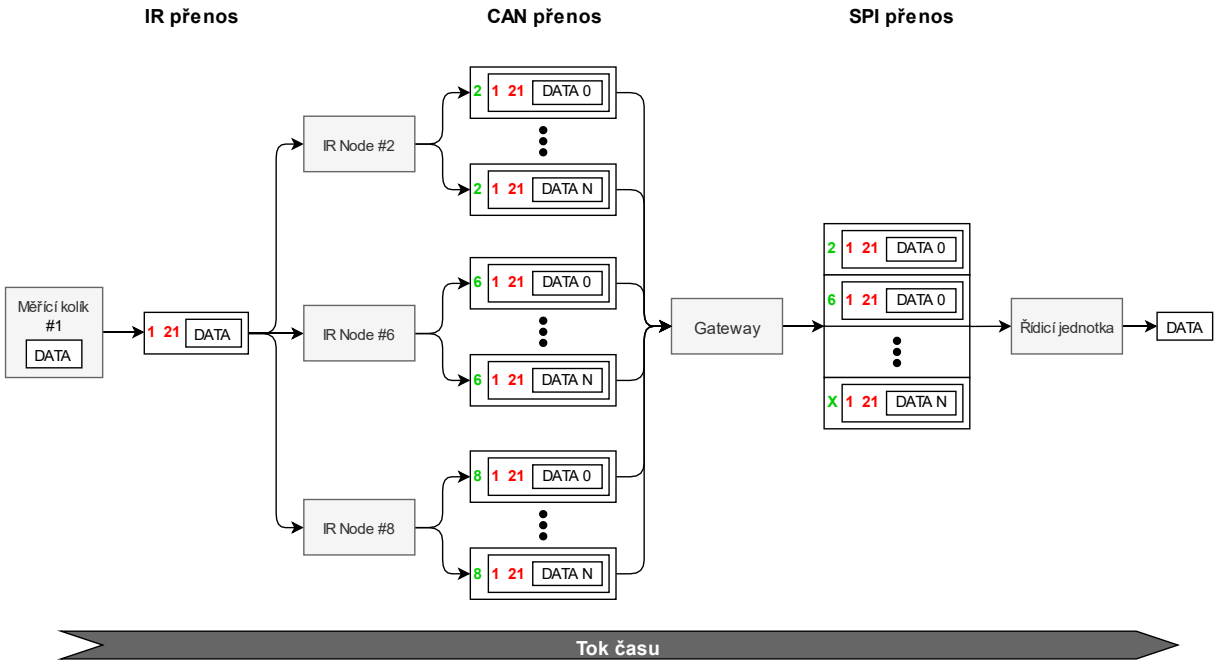
Pro znázornění je níže uveden kontextový diagram datových toků (Obr. 18), kde je schematicky zobrazena komunikace mezi zařízeními v měřicím systému. V diagramu záměrně není specifikován proces symbolizující modelovaný systém, ale pouze terminátory a toky mezi nimi.



Obr. 18 - Kontextový diagram datových toků

Je dobré si povšimnout, že data zasílaná k měřicímu kolíku jsou zaznačena ze zařízení gateway a data odesílaná z měřicího kolíku směřují k IR nodu. Fyzicky dokáže vysílat data pomocí infračerveného přenosu pouze měřicí kolík nebo IR node, zařízení gateway nikoli. Vysvětlením tohoto značení je, že sběrnice CAN je multiplexovaná s IR vysíláním, což znamená, že přenos IR dat je zajištěn fyzickou vrstvou CAN sběrnice. Zařízení gateway má tedy znalost o datech, které pomocí IR přenosu zasílá směrem k měřicím kolíkům, zatímco IR nody pouze hardwarově zprostředkují toto vysílání. Technicky to znamená, že IR nody povolí IR budiče a deaktivují svou CAN periferii, čímž je umožněno přenášet skrze CAN sběrnici data, která nesplňují standardy CAN specifikace.

Návrh komunikačních protokolů pro jednotlivé komunikační prostředky vychází z celkové koncepce přenosu dat v měřicím systému. Klíčovou doménou je přenos dat mezi řídicí jednotkou a měřicími kolíky a mezi řídicí jednotkou a IR nody. Na následujícím obrázku (Obr. 19) je znázorněn modelový příklad toku dat z měřicího kolíku směrem k řídicí jednotce. Vizualizace je ilustrační a slouží jako první krok pro návrh jednotlivých protokolů, zohledňuje však základní principy pro komunikaci, jako je adresování zařízení, užívání příkazů či maximální datovou délku paketu CAN protokolu. Nutno zdůraznit, že komunikace s měřicími kolíky je obousměrná (což ilustrace nezobrazuje), tedy je nutné uvažovat, že měřicí kolík byl před zasláním dat k tomuto vyzván. Příklad uvažuje zaslání dat z měřicího kolíku, který přijmou 3 IR nody.



Obr. 19 - Ilustrační obrázek přenosu dat z měřicího kolíku do řídicí jednotky

Na základě výše uvedené ilustrace lze odvodit některé základní myšlenky ovlivňující návrh jednotlivých komunikačních protokolů. Prvním bodem je skutečnost, že datový tok z měřicího kolíku může být v měřicím systému zachycen předem neznámým počtem IR nodů. Pro předání dat stačí, aby na sběrnici CAN tato data vysílal jen jeden IR node. Každé další vysílání stejných dat jiným IR nodem je redundantní. Komunikace tedy může probíhat různými scénáři:

a) Všechny IR nody se budou snažit přijatá data vyslat dále na sběrnici CAN.

b) IR nody vyšlou zprávu, že zachytily data a zařízení gateway vyzve zvolený IR node, který bude data vysílat.

c) Začne-li jeden IR node vysílat data, ostatní ověří tato data a v případě shody je nezašlou.

Pro řešení byla zvolena možnost a), tedy aby každý IR node vyslal přijatá data od měřicího kolíku dále na sběrnici CAN. Důvodem je, že tato varianta je nejjednodušší na realizaci, a také nejvíce přímočará. Další možnosti jsou sice sofistikovanější, avšak za cenu větší režie při komunikaci nebo přidání dalších funkcionalit zařízením (např. bod c by znamenal přidání kontroly zpráv od jiných IR nodů).

Dále je důležité uvažovat, že data z měřicího kolíku delší než 8 bajtů musí být pro přenos na sběrnici CAN rozloženy do více zpráv, jelikož specifikace CAN neumožňuje přenos většího množství dat v jedné zprávě. V návrhu komunikačního protokolu pro předání dat z měřicího kolíku na sběrnici CAN tedy musí být zaveden indikační parametr, pomocí kterého bude umožněno data zpět z jednotlivých zpráv poskládat ve správném pořadí.

Další myšlenkou, která byla v návaznosti na zvolený způsob předávání dat z měřicího kolíku na sběrnici CAN při návrhu komunikačních protokolů zvážena, je arbitráž na sběrnici CAN. Lze očekávat, že jakmile měřicí kolík vyšle poslední datový bajt své zprávy, kterou se podaří zachytit více

IR nody, dojde k situaci, kdy každý z IR nodů bude chtít tato data předat dále na CAN sběrnici. Dle této úvahy lze očekávat, že bude docházet ke kolizím a bude prováděna arbitráž dle specifikace CAN. Kompletně byl princip arbitráže rozebrán v podkapitole 2.2.2. Technicky to pro návrh komunikačního protokolu znamená vhodnou volbu parametrů obsažených v identifikátoru zprávy, který je při řešení kolizí na sběrnici CAN klíčový.

V případě, že všechna data směřují do řídicí jednotky přes zařízení gateway, vyvstává otázka, ve kterém zařízení provádět rekonstrukci dat. Tím je myšleno skládání dat od měřicích kolíků, které byly na sběrnici CAN rozloženy do více zpráv. Vzhledem k faktu, že řídicí jednotka má vyšší výpočetní výkon než zařízení gateway, byla tato funkce přenechána řídicí jednotce. Pro návrh komunikačního protokolu to znamená, že veškeré zprávy, které zařízení gateway obdrží od IR nodů přes sběrnici CAN, ponechá bez jakékoliv kontroly a úpravy a pouze je předá přes sběrnici SPI řídicí jednotce, která s nimi bude dále pracovat.

Posledním bodem k uvážení při návrhu komunikace je situace, kdy řídicí jednotka nebude číst data ze zařízení gateway dostatečně rychle, což vede k nutnosti řešit hromadění těchto dat. Nejprve tedy dojde k ukládání dat do zařízení gateway, jenže nebude-li řídicí jednotka číst data delší dobu, dojde k vyčerpání paměti v zařízení gateway a každá nově příchozí data budou muset být zahozena. Řešením je využití IR nodů, které mohou uchovávat data z měřicích kolíků. Konkrétně tedy v případě, že v zařízení gateway dojde k překročení určitého limitu obsazenosti paměti, vyšle všem IR nodům pokyn, aby data dále nezasílaly a ukládaly je do paměti. Jakmile řídicí jednotka začne data ze zařízení gateway číst a dojde k uvolnění paměti, je IR nodům zaslána zpráva, která je informuje, že mohou data opět posílat. Popsaným principem sice nelze zcela vyřešit situaci, kdy řídicí jednotka nebude data ze zařízení gateway číst, avšak lze navýšit množství dat, které bude v tomto případě zachráněno. Komunikační protokol na sběrnici CAN musí obsahovat podporu pro zastavení přenosu dat z IR nodů směrem k zařízení gateway aniž by tuto komunikaci musela řídit řídicí jednotka.

4.1 Komunikační protokol pro IR přenos

IR přenos je v realizovaném systému použit pro komunikaci s měřicími kolíky. Komunikační protokol umožňuje konfiguraci a kalibraci měřicích kolíků, spouštění měřicích módů, přenos naměřených dat a speciální funkce, jako je například zvuková signalizace či zjištění stavu baterie.

Pro přenos dat byla vytvořena struktura paketu, která se skládá z hlavičky, datové části a kontrolního součtu. Tato struktura je uvedena v následujících tabulkách.

Tabulka 2: Rámec komunikačního protokolu pro IR přenos směrovaný k měřicímu kolíku

Příchozí zpráva pro měřicí kolík							
Hlavička		**Data**				**Kontrolní součet**	
8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
Pin adr	PCmd	Data	Data	…	Data	Hi	Low

Tabulka 3: Rámec komunikačního protokolu pro IR přenos směrovaný z měřicího kolíku

Odchozí zpráva z měřicího kolíku							
Hlavička		**Data**				**Kontrolní součet**	
8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
Pin adr + 128	PCmd	Data	Data	…	Data	Hi	Low

Hlavička rámce obsahuje 2 bajty. První bajt určuje adresu měřicího kolíku (*Pin adr*). Jedná-li se o zprávu, kterou vysílá měřicí kolík, je k poli s adresou přičtena hodnota 128. Je předpokládáno, že adresa každého kolíku v měřicím systému bude unikátní a tím bude jasně dáno, kterému měřicímu kolíku je zpráva adresována. Distribuce adres je provedena následujícím způsobem:

- 0 – Výchozí adresa pro nový měřicí kolík.
- 1 … 120 – Dostupné adresy pro měřicí kolíky v systému.
- 122 – Broadcast adresa pro měřicí kolíky.

Druhý bajt v hlavičce zprávy je označen jako identifikátor příkazu (*PCmd*). Jedná se o jednoznačnou identifikaci akce, kterou má měřicí kolík vykonat.

Délka datové části paketu není pevně určena, a dokonce není ani vysílána ve zprávě. Z toho plyne, že každá zpráva rozlišená pomocí identifikátoru příkazu (*PCmd*) musí mít pevně stanovenou délku. Tím pádem jakékoliv zařízení komunikující dle popisovaného protokolu musí znát délku každé zprávy. Výhodou tohoto řešení je, že se nemusí přenášet znaky určující začátek a konec zprávy, případně datová délka, čímž dochází k časové úspoře při přenosu dat. Ta je velice vítaná, neboť přenosová rychlost dat přeš IR je několikanásobně nižší, než přenosová rychlost sběrnice CAN či SPI. V podstatě se jedná o nejpomalejší část v celém měřicím řetězci. Nevýhodou však je právě nutnost zavedení délky zpráv do každého přijímacího zařízení.

Pro zabezpečení správnosti přenesených dat je využit kontrolní součet. Ten je implementován jako aritmetický součet všech vysílaných bajtů ve zprávě, tedy součet všech bajtů v hlavičce a datové části. Toto zabezpečení není tak robustní jako například CRC kontrola, neboť kontrolní součet může být stejný pro více zpráv (např. zpráva dekadicky zapsána jako 15 30 15 bude mít stejný součet jako zpráva 30 29 1), nicméně situace, kdy dojde k rozdílnosti mezi vyslanými a přijatými daty při stejném kontrolním součtu nastane velmi ojediněle.

4.2 Komunikační protokol na sběrnici CAN

Pro komunikaci mezi zařízením gateway a IR nody je využito sběrnice CAN. Komunikační protokol byl vytvořen pro strukturu rámců verze 2.0B, kde identifikátor rámce má velikost 29 bitů a je označován zkratkou EID (Extended IDentifier). Aplikační vrstva komunikačního protokolu zajišťuje podporu pro přenos dat z měřicích kolíků, konfiguraci IR nodu a dočasnou deaktivaci CAN rozhraní v IR nodu.

Jelikož jeden rámec dle CAN specifikace dokáže přenést pouze 8 B dat, což pro některé zprávy od měřicího kolíku, jako je například přenos naměřených dat, nestačí, je potřeba každý datový rámec opatřit pořadovým číslem. Pomocí tohoto značení je možné větší datový blok složit zpět z jednotlivých zpráv dohromady.

V následující tabulce (Tabulka 4) je uveden formát paketu pro přenos dat z IR nodu do zařízení gateway. Identifikátor rámce (EID) obsahuje následující údaje:

- Tr adr – Adresa vysílajícího IR Nodu.
- Pin adr – Adresa měřicího kolíku.
- PCmd – Identifikátor příkazu pro měřicí kolík.
- Cnt – Pořadové číslo rámce.

Tabulka 4: Rámec komunikačního protokolu na sběrnici CAN směrovaný pro zařízení gateway

IR node → Gateway											
EID				**DATA**							
5bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
Tr adr	Pin adr	PCmd	Cnt	Data	Data	…	…	…	…	…	Data

Pro přenos dat ze zařízení gateway do IR nodu je datový paket navržen jinak. Jelikož již nedochází k přenášení informací spojených s měřicími kolíky, jsou pole *Pin adr* a *PCmd* vypuštěna. Paket obsahuje tato pole:

- Tr adr – Adresa vysílajícího = Gateway.
- Node adr – Adresa příjemce = konkrétní IR node.
- NCmd – Identifikátor příkazu pro IR node.

Tabulka 5: Rámec komunikačního protokolu na sběrnici CAN směrovaný pro IR node

Gateway → IR Node											
EID				**DATA**							
5bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
Tr adr	Node adr	-	-	NCmd	Data	Data	…	…	…	…	Data

Prvním údajem uvedeným v identifikátoru zprávy dle komunikačního protokolu je adresa odesílatele zprávy (*Tr adr*), a to z důvodu arbitráže. Při dodržení pravidla, že přidělená adresa každému zařízení připojenému ke sběrnici CAN bude unikátní, dojde v případě kolize při vysílání zpráv na sběrnici k vyřešení vzniklé kolize dle specifikace CAN tak, že zařízení s nejnižší adresou bude mít nejvyšší prioritu a další zařízení přestanou vysílat. Tím pádem dojde k vyřešení situace maximálně při odeslání pátého bitu zprávy. Ostatní údaje identifikátoru zprávy zasílané z IR nodů (*Pin adr, PCmd, Cnt*) mohou být pro kolidující zprávy totožné, a tudíž na jejich základě nelze provádět arbitráž. Z toho také vyplývá, že pravidlo pro unikátní přidělení adres zařízením na sběrnici CAN musí být striktně dodrženo, jinak by mohla zařízení se stejnou adresou pokračovat ve vysílání celého identifikátoru (EID) a tím způsobit chybu.

K zabezpečení správnosti přenosu dat není potřeba do komunikačního protokolu zavádět jakékoliv ochranné prvky, jelikož tato funkcionalita je řešena samotnou specifikací CAN rámce v podobě CRC kontroly.

Jak již bylo zmíněno, adresace zařízení na sběrnici CAN je prováděna prvními 5 bity v identifikátoru zprávy, což znamená, že maximální počet přiřaditelných adres je 32 (2^5). Distribuce adres je rozvržena následujícím způsobem:

- 0 – Adresa pro zařízení gateway.
- 1 … 29 – Adresa pro IR nody.
- 30 – Výchozí adresa pro nový IR node.
- 31 – Broadcast adresa pro IR nody.

Z této distribuce adres vyplývá, že maximální počet unikátně adresovatelných zařízení na sběrnici je 31. Nejvyšší adresa (31) je vyhrazena pro zaslání zprávy všem IR nodům na sběrnici. Adresa pro zařízení

gateway je zvolena 0 z toho důvodu, že při souběžném vysílání více zařízení na sběrnici CAN má zařízení s nejnižší adresou nejvyšší prioritu.

4.3 Komunikační protokol na sběrnici SPI

Sběrnice SPI slouží v realizovaném měřicím systému pro přenos dat mezi zařízením gateway a řídicí jednotkou. Komunikace je zavedena modelem Master/Slave, kdy řídicí jednotka zaujímá Master roli, čímž sama řídí výměnu dat. Komunikační protokol zprostředkuje řídicí jednotce přenos dat na sběrnici CAN nebo na IR vysílání, umožňuje konfiguraci zařízení gateway a také umožňuje předání dat, které zařízení gateway obdržela od IR nodů.

Struktura paketu pro přenos dat není pevně stanovena, odvíjí se však od účelu komunikace mezi zařízeními. Jedinou podmínkou komunikačního protokolu je, aby první datový bajt, který zasílá řídicí jednotka (Master) do zařízení gateway (Slave), byl identifikátor konkrétní akce. Struktura předávání dalších dat je na základě tohoto identifikátoru vytvořena samostatně. Pro zabezpečení správnosti přenosu dat je využit kontrolní součet, stejně jako tomu je u komunikace s měřicími kolíky.

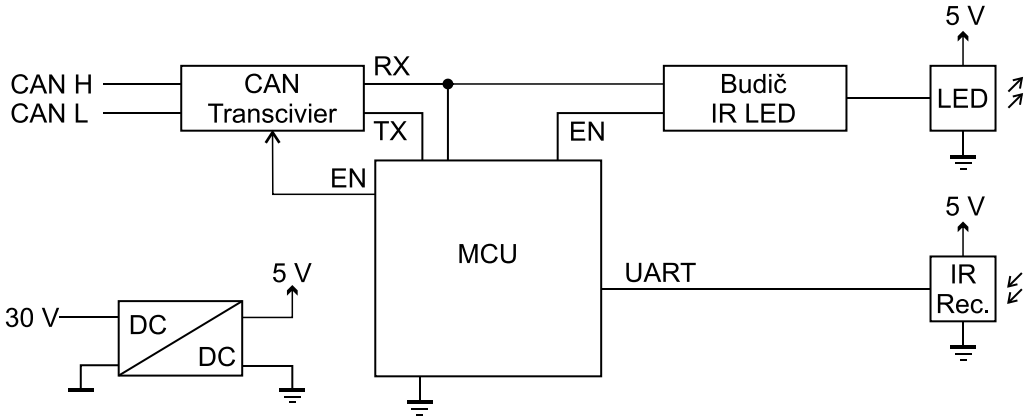
V níže uvedeném obrázku tabulky z nástroje Excel jsou vybrány konkrétní příklady výměny dat, které byly mezi řídicí jednotkou (Master) a zařízením gateway (Slave) implementovány. Lze pozorovat popsanou skutečnost, že komunikaci zahajuje řídicí jednotka identifikátorem příkazu, od kterého se odvíjí další zasílání dat. V případě příkazu, kdy je dotázáno zařízení gateway, zda má k dispozici data k předání, není proveden kontrolní součet, neboť kontrola jednoho datového bajtu zcela postrádá smysl.

VÝZNAM		PŘÍKAZ	DATA						Kontrolní součet	
		8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
Poslat zprávu na sběrnici CAN	MOSI	**Pošli data na CAN** *0x10*	Délka dat (N)	NodeAdr	Data[0]	Data[1]	Data[...]	Data[N-2]	Hi	Lo
	MISO	-	-	-	-	-	-	-		-
Poslat zprávu skrze IR přenos	MOSI	**Pošli IR data** *0x20*	Délka dat (N)	PinAdr	Data[0]	Data[1]	Data[...]	Data[N-2]	Hi	Lo
	MISO	-	-	-	-	-	-	-		-
Jsou dostupná data k předání?	MOSI	**Jsou data?** *0x70*	-	-	-	-	-	-		-
	MISO	-	Ano / Ne *0x71 / 0x74*	-	-	-	-	-		-
Změna rychlosti CAN sběrnice	MOSI	**Změna Can rychlosti** *0x40*	ryhlost CAN *0x12/25/50/A0*	-	-	-	-	-	Hi	Lo
	MISO	-	-	-	-	-	-	-		-

Obr. 20 – Příklady komunikace mezi řídicí jednotkou a zařízením gateway přes sběrnici SPI

5 Návrh a tvorba firmware pro zařízení IR node

Jak již bylo popsáno v první kapitole, IR node slouží v měřicím řetězci pro příjem dat z měřicích kolíků, které předává dále na sběrnici CAN, a k fyzickému vysílání IR dat směrem k měřicím kolíkům. Na následujícím obrázku (Obr. 21) je zobrazeno blokové schéma elektroniky IR nodu.



Obr. 21 - Blokové schéma elektroniky IR nodu

Konkrétním mikrokontrolerem, který byl v rámci návrhu IR nodu použit, je typ PIC18F26K80 od společnosti Microchip Technology. Jedná se o 8bitový MCU vybavený integrovanou CAN periferií, která je pro zařízení IR node klíčová. Frekvenci hodinového signálu procesoru lze nastavit až na 64 MHz. Výčet dalších parametrů MCU je uveden v následujícím obrázku.

Device	Program Memory	Data Memory (Bytes)	Data EE (Bytes)	Pins	I/O	CTMU	12-Bit A/D Channels	CCP/ ECCP	Timers 8-Bit/16-Bit	EUSART	Comparators	ECAN™	MSSP	BORMV/LVD	DSM
PIC18F26K80	64 Kbytes	3,648	1,024	28	24	1	8-ch	4/1	2/3	2	2	1	1	Yes	No

Obr. 22 - Parametry mikrokontroleru PIC18F26K80

Pro realizaci vlastního firmwaru pro MCU bylo potřeba využít dostupných vývojářských nástrojů. Výrobce použitého MCU, společnost Microchip, nabízí spousty softwarových a hardwarových prostředků, mnohé z nich jsou dokonce dostupné zdarma. Z kategorie softwarových nástrojů lze využít vývojová prostředí pro tvorbu kódu, kompilátory, konfigurační a produkční systémy. Hardwarovými nástroji jsou programátory a debuggery, které umožňují nahrání uživatelského firmwaru do MCU a jeho následné odladění. Výběr konkrétních vývojářských nástrojů závisí zejména na rodině MCU, pro který je požadováno firmware vyvíjet. [16]

Konkrétními nástroji vybranými pro tuto práci bylo vývojové prostředí MPLAB X IDE. Jediným možným kompilátorem pro rodinu PIC MCU je MPLAB XC, přičemž vzhledem k tomu, že IR node využívá 8bitový MCU, padl výběr konkrétní verze kompilátoru na MPLAB XC8. Nástrojem pro nahrání a odladění firmwaru byl zvolen MPLAB PICkit3.

5.1 Návrh architektury firmwaru

Prvním krokem pro návrh architektury firmwaru bylo provedení analýzy funkčních požadavků kladených na zařízení. Z funkcionality IR nodu v měřicím systému lze určit, že mezi hlavní funkční požadavky na firmware IR nodu se řadí:

- Příjem bajtů na UART rozhraní (tj. bajty přijaté skrze IR přenos).
- Kontrola komunikačního protokolu IR přenosu.
- Komunikace na sběrnici CAN.
- Povolení budiče pro IR vysílání.

Aby bylo dosaženo co nejrychlejší odezvy na důležité asynchronní události, byla v návrhu firmwaru využita přerušení. Ta byla navíc rozdělena do dvou úrovní pomocí priorit, které je možno v mikrokontroleru nastavit. Tím došlo k rozdělení programu do třech úrovní, kdy nejnižší úroveň je hlavní smyčka vykonávající periodickou činnost, nad touto úrovní jsou přerušení nízké priority a nejvyšší úrovní jsou přerušení vysoké priority. Dojde-li tedy ke vzniku události, která má vyšší úroveň než právě vykonávaný kód, je vykonávání jednoduše řečeno přerušeno, vykoná se obsluha vzniklé události (přerušení) a následně se pokračuje ve vykonávání předchozích instrukcí. Ve firmwaru byla využita tato přerušení:

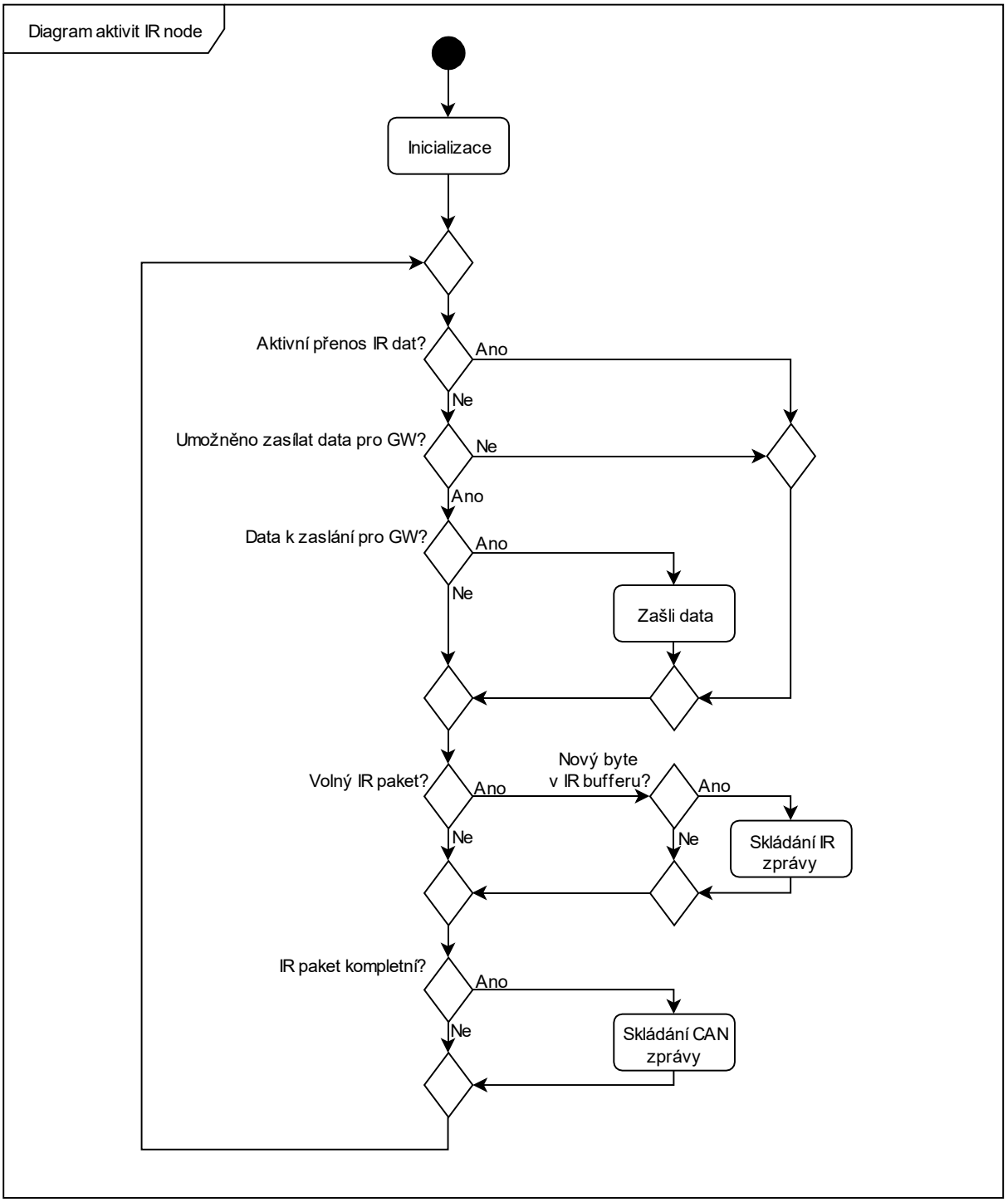
- S vysokou prioritou: UART Rx, 1x časovač.
- S nízkou prioritou: CAN Tx, CAN Rx, 2x časovač.

Nejvyšší úroveň má kód pro obsluhu přijatých bajtů na UART rozhraní, tedy příjem bajtů zasílaných od měřicích kolíků IR přenosem. Důvodem je, že při ztrátě jediného bajtu by došlo ke ztrátě celé zprávy, neboť by neseděla délka dat či kontrolní součet. Dále má nejvyšší prioritu přiřazenou časovač, který je využit k časování doby pro povolení budiče IR LED a s tím souvisejícím vypnutím CAN periferie, což slouží pro umožnění vysílání IR dat zařízením gateway. Důvodem je, aby tato doba, která je zasílaná ve zprávě od zařízení gateway, byla dodržena s co největší přesností. Obslužné rutiny těchto asynchronních událostí mají co možná nejméně instrukcí, aby jejich vykonávání zbytečně nebránilo ve vykonávání událostí nižších úrovní či jiných částí programu.

Nižší prioritu má obslužný kód pro komunikaci na sběrnici CAN. V případě přijetí zprávy je okamžitě provedeno zpracování zprávy a vykonání požadované akce (je-li pro IR node známá). Událost označená jako CAN Tx slouží pro indikaci, že zpráva, která byla předána na odeslání byla fyzicky opravdu odeslána. Dále jsou použity také dva časovače. První časovač slouží pro signalizaci překročení časového limitu mezi přijetím dvou bajtů na rozhraní UART. Jeho účel je více rozveden dále v podkapitole 5.2. Druhý časovač slouží jako signalizace, že uplynul pevně nastavený časový úsek, po který IR node nevyslal žádnou zprávu na sběrnici CAN, a tedy že by měl zaslat tzv. Alive zprávu. Tato zpráva je směřována k zařízení gateway a slouží k indikaci, že dané zařízení je schopno komunikovat.

Nejnižší úroveň je přiřazena hlavní smyčce programu, ve které jsou prováděné všechny ostatní operace. Postup vykonávání hlavní smyčky je znázorněn pomocí UML diagramu aktivit (Obr. 23). Při vstupu do hlavní funkce (main) dojde nejprve k inicializaci, což je aktivita, při níž jsou vybrány a nastaveny periferie MCU, které jsou v programu využívány, dále jsou načteny konfigurační parametry z paměti EEPROM (případně nastaveny výchozí hodnoty parametrů nejsou-li v paměti uložené), nastaveny jednotlivé priority přerušení, a nakonec také povoleno globální využití přerušení. Následně

program vstoupí do nekonečné smyčky, kde jsou prováděny všechny periodické činnosti, jako je skládání IR zprávy ze sekvence přijatých bajtů skrze IR přenos, následné rozkládání IR zprávy do korespondujících CAN zpráv atd.



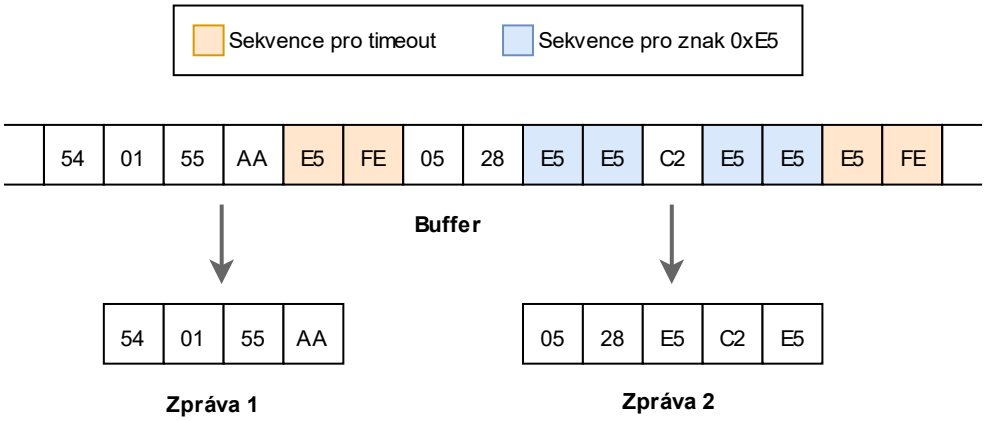
Obr. 23 - UML diagram aktivit hlavní funkce firmwaru IR nodu

5.2 Příjem IR dat

Jak již bylo zmíněno, jednou z hlavních funkcionalit IR nodu je příjem dat od měřicích kolíků a předání těchto dat dále na sběrnici CAN. Každý přenesený bajt z měřicích kolíků je dekódován IR přijímačem a předán na rozhraní UART do mikrokontroleru. Tato data jsou následně v programu ukládána do kruhového bufferu. Jelikož zápis dat do bufferu se děje v přerušení a čtení dat z bufferu je činností hlavní smyčky programu, bylo nutné vzít v potaz situaci, kdy zápis může přerušit čtení v jakémkoli kroku. Z toho důvodu bylo nutné přizpůsobit samotnou implementaci bufferu ve zdrojovém kódu programu tak, aby nemohla být narušena integrita dat.

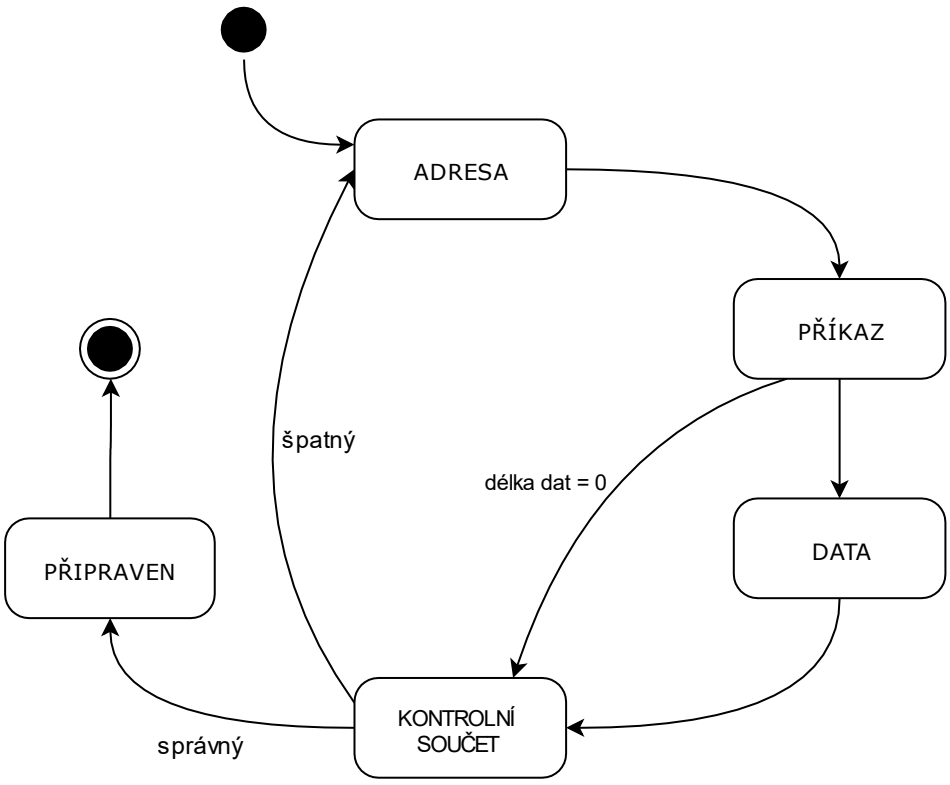
Zprávy zasílané skrze IR přenos nemají dle vytvořeného komunikačního protokolu vyhrazen speciální znak pro indikaci začátku a konce, avšak lze je rozlišit časovým limitem (timeoutem). Před každým zasláním zprávy má měřicí kolík nastavenou synchronizační mezeru 8 ms. Při přijmu datového bajtu je v IR nodu spuštěn časovač, který v případě dosažení hodnoty 8 ms indikuje vypršení časového limitu pro příjem dalšího datového bajtu od měřicího kolíku.

V bufferu, do kterého se bajty přijaté skrze IR přenos ukládají, je nutné indikovat, že došlo k vypršení časového limitu pro přenos zprávy, a tedy se jedná o konec zprávy. Pro tuto situaci byla vytvořena tzv. úniková sekvence (anglicky známá jako escape sequence). Jedná se o dvojici po sobě jdoucích bajtů, které při vložení do bufferu oddělují dvě různé zprávy. Konkrétně byla zvolena dvojice bajtů hexadecimálně zapsaných jako 0xE5 a 0xFE. První ze znaků je tzv. únikový znak (escape character). Jelikož tento znak může obsahovat také samotná zpráva zasílaná měřicím kolíkem, je nutné vytvořit únikovou sekvenci rovněž pro tento znak. Tato sekvence byla vytvořena zdvojením znaku, tedy bajt přijatý jako 0xE5 je do bufferu vložen jako sekvence 0xE5 a 0xE5. Celý výše popsaný princip je znázorněn na následujícím obrázku.



Obr. 24 - Princip escapování znaků v bufferu

Při vyčítání dat z bufferu je rekonstruována přijatá zpráva a kontrolován komunikační protokol. Kontrola je realizována stavovým automatem (Obr. 25), kde jednotlivé stavy korespondují s částmi definovanými dle protokolu. Pokud je v jakémkoli stavu z bufferu vyčtena sekvence označující timeout zprávy, je skládání zprávy ukončeno a proces začíná opět od prvního stavu. Tato situaci není v níže uvedeném obrázku znázorněna.



Obr. 25 - Stavový automat pro kontrolu komunikačního protokolu IR přenosu

Jakmile je zpráva úspěšně rekonstruována (stavový automat se nachází ve stavu připraven), dojde k jejímu postupnému rozložení do zpráv pro komunikaci po sběrnici CAN. Ty jsou postupně odesílány a v momentě odeslání poslední zprávy je stavový automat ukončen a paměť využitá pro uchování IR zprávy je uvolněna.

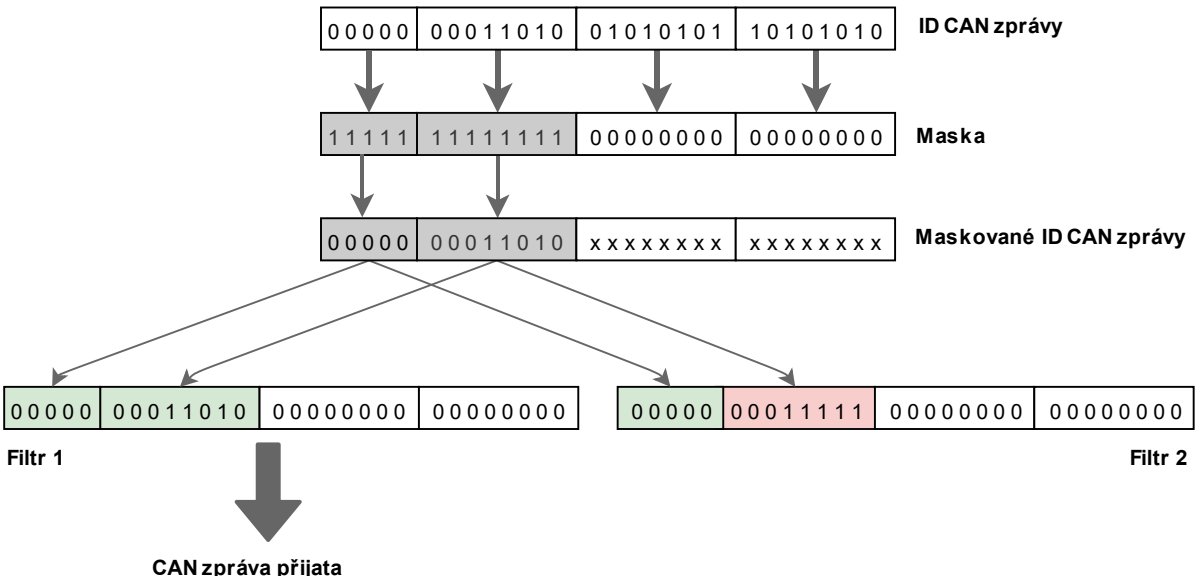
5.3 Komunikace na sběrnici CAN

IR node rozeznává od zařízení gateway příkazy pro nastavení konfiguračních parametrů, mezi které patří přenosová rychlost UART rozhraní (ta se odvíjí od přenosové rychlosti IR dat), přenosová rychlost CAN sběrnice a adresa zařízení, dále také příkaz pro deaktivaci CAN periferie v MCU na stanovenou dobu a příkaz pro zastavení zasílání dat na sběrnici CAN.

Z koncepce měřicího řetězce vyplývá, že IR nodů bude na sběrnici CAN připojených hned několik (maximálně 30). IR nody však mezi sebou nekomunikují. Jediné zprávy, které zpracovávají, mohou přicházet od zařízení gateway. Z návrhu komunikačního protokolu na sběrnici CAN (kapitola 4.2) je patrné, že tyto zprávy lze rozlišit pomocí adresy odesílatele, kterým může být pouze zařízení gateway s přidělenou adresou 0, a adresy příjemce, což je konkrétní IR node.

Funkcionalitu, zda zpráva patří danému IR nodu lze řešit softwarově, kdy u každé přijaté zprávy proběhne kontrola adresy odesílatele a adresy příjemce zprávy a následně se vyhodnotí, zda zpracovat obsah zprávy či ne. Toto řešení má však velkou nevýhodu. Převážná část komunikace na sběrnici CAN je směrována od IR nodů k zařízení gateway a pouze nepatrná část od zařízení gateway ke konkrétnímu IR nodu. Firmware IR nodu by se s každou zprávou na sběrnici CAN dostal do přerušení, kde by teprve provedl rozhodnutí o zpracování obsahu zprávy a ve většině situací by bylo zjištěno, že zpráva není pro něj. Když se toto skloubí se situací, kdy například 20 IR nodů zachytí zprávu přes IR přenos od měřicího kolíku a bude ji chtít poslat dále po sběrnici CAN, dojde k situaci, kdy každý IR node 19x přeruší svou činnost kvůli zprávě, která není pro něj. A tato situace se může zopakovat několikrát po sobě, neboť data z měřicího kolíku mohou být rozložena do několika CAN zpráv.

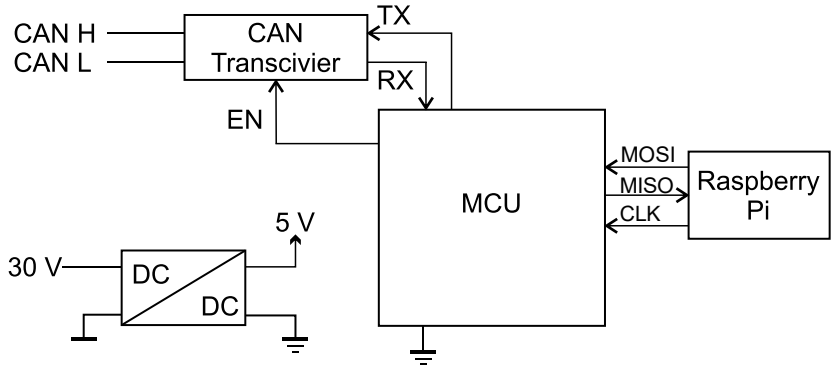
Mnohem lepším řešením, které bylo také při tvorbě firmwaru využito, je provést hardwarovou filtraci CAN zpráv. Tu lze realizovat pro periferii CAN nastavením filtrů a masek, které zajistí, že se do MCU dostanou pouze ty zprávy, které vyhovují nastaveným kritériím. Filtrační kritéria lze nastavit pouze pro identifikátor zprávy (ID). V případě IR nodu je potřeba nastavit masku na část identifikátoru, kde je uvedena adresa odesílatele a adresa příjemce. Konkrétními hodnotami identifikátoru jsou adresa zařízení gateway (0) a adresa daného IR nodu (1-30). Popsaný princip filtrace CAN zpráv je uveden v obrázku níže (Obr. 26). Nelze však zapomenout, že IR nody mohou být adresovány také broadcast adresou (31), tudíž bylo potřeba nastavit druhý filtr pro tento účel. Nastavení filtrů probíhá při inicializaci a jelikož adresa daného zařízení může být pomocí CAN zprávy změněna i při jeho běhu, je důležité zároveň změnit hodnotu jeho CAN filtru.



Obr. 26 - Princip hardwarové filtrace CAN zpráv

6 Návrh a tvorba firmware pro zařízení gateway

Jak bylo již dříve zmíněno, zařízení gateway je bránou mezi řídicí jednotkou a ostatními zařízeními v měřicím řetězci. Zařízení je připojeno na sběrnici CAN a s řídicí jednotkou komunikuje pomocí sběrnice SPI. Na níže uvedeném obrázku je zachyceno blokové schéma elektroniky tohoto zařízení.



Obr. 27 - Blokové schéma elektroniky zařízení gateway

Elektronika zařízení gateway je řízena úplně stejným typem mikrokontroleru (PIC18F26K80 od společnosti Microchip Technology), který byl použit pro IR node. Popis parametrů mikrokontroleru a nástroje použité pro realizaci firmwaru lze nalézt na první stránce předcházející kapitoly pojednávající o IR nodu.

6.1 Návrh architektury firmwaru

Prvním krokem při návrhu firmwaru pro zařízení gateway je opět analýza funkčních požadavků kladených na zařízení. Mezi hlavní požadavky patří:

- Komunikace s řídicí jednotkou pomocí SPI.
- Komunikace na sběrnici CAN.
- Vysílání IR dat pomocí fyzické vrstvy sběrnice CAN.
- Probouzení měřicích kolíků.

První tři body funkčních požadavků jsou blíže rozebrány v dalších podkapitolách. Posledním bodem je probouzení měřicích kolíků, což je důležitá funkce pro správnou komunikaci s měřicími kolíky. Každý měřicí kolík je totiž napájen z vlastní baterie a v rámci snížení spotřeby je v něm zavedena funkcionalita, kdy nepobíhá-li po určitou dobu žádný přenos IR dat (jak vysílání, tak příjem z pohledu kolíku), je kolík uspán. Probuzen je v momentě přijetí datového bajtu skrze IR přenos (kolík nemusí být adresován, jde jen o příjem jakéhokoliv bajtu IR přijímačem). Tento bajt však nezpracuje jako součást zprávy, a proto je nutné, aby mu byl zaslán libovolný bajt, kterým je kolík probuzen a následně až zpráva. A jelikož zařízení gateway je tím, kdo IR data pro měřicí kolíky vysílá, byl mu tento funkční požadavek přidělen, jelikož dokáže nejlépe časovat interval, který je pro uspání měřicího kolíku použit. Samozřejmostí je,

že interval, který měří zařízení gateway, by měl být o něco menší než interval, který je použit pro uspání měřicího kolíku v jeho firmwaru. Důvodem je předcházení situaci, kdy zařízení gateway nevyprší časový interval, a tedy si bude myslet, že měřicí kolík uspán není a nepošle mu bajt pro probuzení, avšak měřicí kolík uspán bude. Tím by došlo k neúspěšnému zaslání zprávy. Naopak pokud je bajt pro probuzení zaslán, ale měřicí kolík uspán není, nestane se nic. Tím pádem bylo vhodné toto ošetření implementovat.

Stejně jako u IR nodu tak i zde bylo pro reakci na asynchronní události využito systému přerušení a rozdělení jednotlivých zdrojů přerušení do dvou priorit, čímž je program rozdělen do třech úrovní. Pro zařízení gateway byla využita tato přerušení:

- S vysokou prioritou: SPI, 1x časovač.
- S nízkou prioritou: CAN Tx, CAN Rx, 1x časovač.

Nejvyšší priorita je přiřazena komunikaci s řídicí jednotkou po sběrnici SPI. Důvodem je, že mikrokontroler disponuje pouze jedním 8bitovým bufferem a při přijetí bajtu od řídicí jednotky do bufferu je důležité bajt ve firmwaru co nejrychleji z bufferu převzít, jelikož může dojít k situaci, že řídicí jednotka zašle další bajt, kterým předchozí přepíše. Dále je nejvyšší priorita přerušení přidělena časovači, který slouží pro softwarové generování IR vysílání. U generování IR je žádoucí, aby byla co nejpřesněji dodržena frekvence signálu, z toho důvodu má časovač nejvyšší prioritu. Princip vysílání IR dat je detailně popsán v následující podkapitole 6.2.

Nižší prioritu událostí má komunikace na sběrnici CAN. Příchozí zprávy, což jsou zprávy od IR nodů, jsou ukládány do bufferu pro předání řídicí jednotce. Přerušení při odchozí zprávě značí pouze situaci, že zpráva předaná k odeslání byla skutečně fyzicky odeslána. Dále má nižší prioritu přidělenou časovač, kterým je měřen interval určující, zda mají být měřicí kolíky probuzeny před IR vysíláním. V přerušení dochází pouze k změně proměnné, která tuto informaci nese.

Nejnižší úroveň priority vykonávání programu má hlavní smyčka. Při vstupu do funkce main dojde v prvním kroku k inicializaci. Při této činnosti dojde k nastavení potřebných periférií MCU, načtení konfiguračních parametrů z paměti EEPROM, případně nastavení výchozích konfiguračních parametrů, a nakonec dojde k nastavení jednotlivých priorit pro přerušení a jejich povolení. V dalším kroku program vstoupí do nekonečné smyčky, kde jsou prováděny jednotlivé rutinní funkce. Zde dochází k ověření obsazenosti příchozího bufferu pro CAN zprávy a v případě překročení limitu obsazenosti nebo naopak při uvolnění bufferu je vyslána zpráva o stavu na sběrnici CAN. Také se zde ověřuje, zda byla obdržena zpráva od řídicí jednotky a v případě že ano, je přikročeno k provedení požadované akce. Tou může být změna konfiguračních parametrů, zaslání přijatých dat na sběrnici CAN či zaslání dat skrze IR přenos.

6.2 Vysílání IR dat

Vysílání IR dat je důležitou funkcionalitou, kterou má za úkol zařízení gateway vykonávat. Jak již bylo dříve popsáno, elektronika zařízení gateway nedisponuje žádnými IR vysílači, ale využívá IR nodů na sběrnici CAN, pomocí nichž je schopna data vysílat.

Prvním důležitým krokem je tedy připravit IR nody tak, aby bylo vysílání IR dat skrze sběrnici CAN umožněno. Tento proces začíná vysláním příslušné CAN zprávy všem IR nodům pomocí broadcast adresy, ve které je také uveden časový limit v milisekundách, po který má být IR vysílání povoleno. Výpočet této doby je proveden na základě následujícího vzorce:

$$t_{IR} = \frac{10^3}{IR_{br}} \cdot WS \cdot DL + t_s\ (ms), \tag{6.1}$$

kde t_{IR} je doba v milisekundách, po kterou bude IR vysílání povoleno, IR_{br} je přenosová rychlost pro IR vysílání, WS je délka (počet bitů) datového slova, DL je počet datových slov ve zprávě a t_s je časový interval v milisekundách přidaný navíc pro bezpečnost.

Druhým krokem zařízení gateway je deaktivace vlastní periferie CAN a nastavení příslušných pinů, aby bylo možno využít fyzickou vrstvu sběrnice CAN pro přenos IR dat. Po tomto kroku je již vše připraveno pro to, aby mohla být IR data přenášena.

Dalším krokem je samotný přenos IR dat. Signál pro přenos dat pomocí IR je modulován softwarově. S odkazem na kapitolu 2.1, signál pro IR přenos se skládá z nosného signálu a modulačního signálu. Nosným signálem je obdélníkový signál o frekvenci 38 kHz a střídě 50 %. Frekvence nosného signálu je pevně určena IR přijímači, kterými jsou vybaveny měřicí kolíky. Modulačním signálem jsou data kódovaná unipolární NRZ metodou. Modulační frekvence je rovna přenosové frekvenci, tedy při nastavení přenosové rychlosti například na 2400 bit/s bude modulační frekvence rovna 2,4 kHz.

Pro softwarovou modulaci dat pro IR přenos je v MCU využit časovač. Ten je nastaven tak, aby generoval přerušení s frekvencí rovnající se dvojnásobku frekvence nosného signálu, tedy s frekvencí 76 kHz. Při tomto přerušení dojde vždy k překlopení logického stavu nosného signálu z 0 do 1 a naopak, čímž je vytvořen obdélníkový signál o frekvenci 38 kHz. Pro generování modulačního signálu je využit stejný časovač. Důvodem je skutečnost, že frekvence modulačního signálu je několikanásobně nižší než frekvence nosného signálu, a tedy je možno poměrně přesně časovat změnu úrovně modulačního signálu. Konstanta pro časování změny úrovně modulačního signálu je vypočtena vztahem:

$$k = \frac{2 \cdot f_c}{f_m} + 0{,}5\ (-), \tag{6.2}$$

kde k je konstanta časovače, f_c je frekvence nosného signálu a f_m je frekvence modulačního signálu. Konstanta časovače je zaokrouhlena na celá čísla a jinými slovy značí kolikrát je přerušení vyvoláno, než dojde k přechodu na další bit datového slova. Přesnějším řešením by samozřejmě bylo využití dvou časovačů, kde jeden by generoval nosný signál a druhý modulační signál, avšak pro účely vývoje bylo využité řešení dostačující.

Jakmile jsou požadovaná IR data vyslána, je příslušný časovač generující modulovaný signál deaktivován. Následně je na zařízení gateway opět aktivována periferie CAN a spuštěn časovač, který se stará o měření časového intervalu pro potřebu probouzení měřicích kolíku před další IR vysíláním.

6.3 Komunikace přes SPI

Pomocí rozhraní SPI komunikuje zařízení gateway s řídicí jednotkou měřicího systému. Prvním krokem byla volba a nastavení parametrů pro periferii na MCU. Dle modelu master/slave je zařízení gateway nastaveno jako slave. Dále byla potřeba nastavit SPI mód, který je standardně definován pomocí kombinace dvou konfiguračních bitů označovaných jako CPOL a CPHA. V mikrokontrolerech PIC jsou tyto konfigurační bity značeny jako CKE a CKP, kde bit CKE odpovídá bitu CPOL a bit CKP je negací bitu CPHA. Rozhraní SPI bylo nastaveno pro mód 0 (CPOL = 0, CPHA = 0).

Důležitým bodem pro návrh firmwaru bylo, aby SPI komunikace byla řešena asynchronně. Není vhodné, aby při komunikaci s řídicí jednotkou byl mikrokontroler zařízení gateway zaneprázdněn pouze výměnou dat. Řešení tohoto bodu vede k využití přerušení. Zároveň není vhodné, aby při vzniku přerušení (tj. v momentě dokončení výměny bajtu s řídicí jednotkou) byla tato činnost přerušena jinou událostí. Z tohoto důvodu je komunikaci skrze rozhraní SPI přiřazena nejvyšší priorita.

Při vzniku přerušení, kdy zařízení gateway předává data řídicí jednotce, je vhodné, aby vyčtení bajtu z bufferu (tj. příchozích dat od řídicí jednotky) a vložení bajtu do buffer pro přenos (tj. odchozích dat do řídicí jednotky) bylo provedeno v co nejméně instrukcích. Důvodem je, že hodinový signál použitého MCU pro zařízení gateway může být maximálně nastaven na 64 MHz a hodinový signál rozhraní SPI se může pohybovat také v řádech MHz (rychlost je určena řídicí jednotkou – masterem). Během několika instrukcí musí být výměna dat provedena, což pro implementaci znamená, že při vzniku přerušení je potřeba bajt okamžitě vyčíst z bufferu do proměnné a předem připravený bajt pro odeslání do bufferu vložit. Po těchto krocích je možno přistoupit k práci s přijatým bajtem, případně přípravě následujícího bajtu pro odeslání při dalším přerušení.

6.4 Komunikace na sběrnici CAN

Zařízení gateway na sběrnici CAN přijímá všechny zprávy, které na ni jiná zařízení vysílají. Žádným způsobem nevaliduje jakoukoliv část datových zpráv, tím pádem potvrzuje a ukládá do svého bufferu i takové zprávy, které nekorespondují s komunikačním protokolem na sběrnici CAN. Kontrola obsahu zpráv je přenesena dále na řídicí jednotku, úkolem zařízení gateway je pouze zprávy přijímat. Z toho plyne, že zařízení gateway nemá nastavenou filtraci zpráv, jako tomu je u IR nodů.

Co se týče vysílání zpráv na sběrnici CAN, samotné zařízení gateway využívá pouze zprávy pro řízení komunikace ve smyslu povolení či zastavení zasílání CAN zpráv od IR nodů a zprávu pro povolení IR vysílání na IR nodech. Jinak zprostředkovává zaslání jakékoliv CAN zprávy ve formátu definovaném v komunikačním protokolu na sběrnici CAN, čehož využívá řídicí jednotka, která definuje obsah zprávy.

7 Návrh a tvorba řídicí aplikace pro centrální jednotku

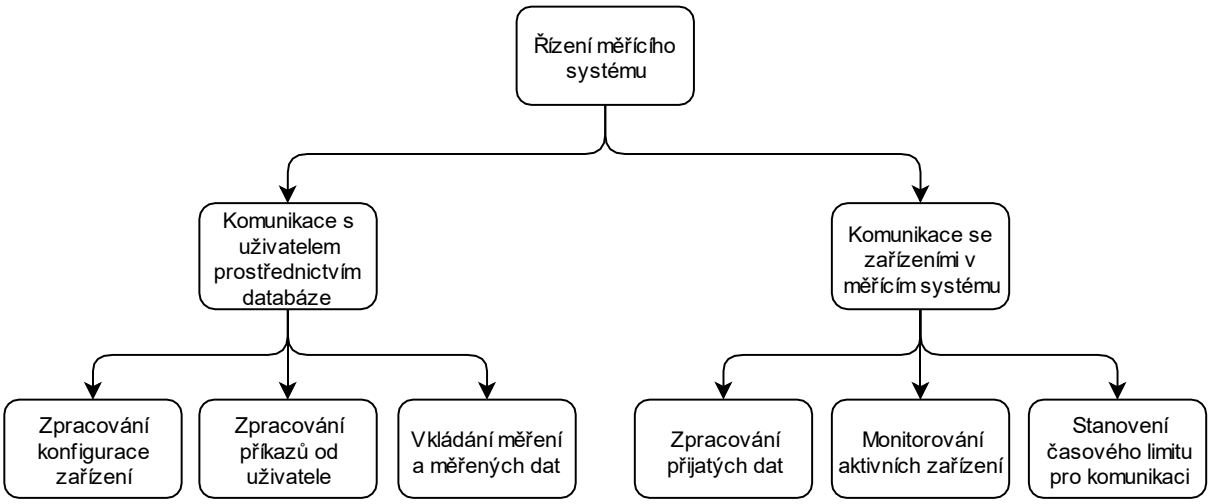
Centrální řídicí jednotkou celého měřicího systému byl zvolen jednodeskový počítač Raspberry Pi, model 3B+ (viz. kapitola 1.4). Jako operační systém jednotky byl vybrán Raspbian. Pro komunikaci mezi centrální jednotkou a celým měřicím řetězcem prostřednictvím zařízení Gateway je využita sběrnice SPI, jejíž piny se nachází na GPIO portu zařízení. Komunikační protokol využitý pro komunikaci na sběrnici SPI je popsán v kapitole 4.3.

Rozhraním pro komunikaci mezi řídicí aplikací a uživatelem je databáze. Konkrétním databázovým prostředkem byla v rámci řešení projektu zvolena relační databáze MySQL, která je nainstalována a provozována přímo v řídicí jednotce. Výhodou komunikace s měřicím systémem prostřednictvím databáze je možnost vytváření dalších aplikací pomocí jakýchkoliv technologií, které umožňují připojení k databázi. Tím pádem lze vytvořit i více frontend aplikací sloužících pro různé účely (např. pro ovládání, pro analýzu měřených dat, pro servisní účely aj.), případně pro různé platformy (např. PC, tablet, dotykový panel aj.).

Jako vhodný programovací jazyk pro vytvoření řídicí aplikace byl vybrán jazyk C, avšak po důkladnější analýze byl zvolen jazyk C++, jehož podmnožinou je původně zvolený jazyk C. Důvodem byla zejména potřeba využití front a listů různých datových typů, na což poskytuje jazyk C++ větší oporu, zároveň umožňuje využívat generičnost. Dále je zde výhodou možnost vytvářet jak procedurální kód, tak také objektově orientovaný kód, což umožňuje větší flexibilitu při vývoji.

7.1 Požadavky na řídicí aplikaci

Prvním krokem při tvorbě řídicí aplikace byl její návrh na základě analýzy funkčních požadavků. Jelikož funkčních požadavků na řídicí aplikaci bylo možno definovat hned několik, bylo je vhodné pro lepší přehlednost rozvrstvit do několika úrovní, kdy každá nižší úroveň zmenšuje míru abstrakce pohledu na realizovaný systém a zvětšuje konkrétnost dané funkce. Výhodou tohoto přístupu je také fakt, že každý funkční požadavek je zpětně vysledovatelný a testovatelný. Na následujícím obrázku (Obr. 28) je uvedeno blokové schéma dekompozice provedené funkční analýzy řídicí aplikace pro první 3 úrovně.



Obr. 28 - Analýza funkčních požadavků

Pod každým funkčním požadavkem v analýze lze uvažovat několik implementačních úkonů spojených s jeho realizací. Pro příklad u funkčního požadavku „*Komunikace s uživatelem prostřednictvím databáze*“ bylo zapotřebí vytvořit v řídicí aplikaci tyto funkcionality:

- Inicializace databáze a vytvoření připojení k databázi.
- Vyčítání dat z databáze.
- Vkládání dat do databáze.
- Provádění změn dat v databázi.
- Ukončení spojení s databází.

Všechny výše uvedené funkcionality musí být v řídicí aplikaci zajištěny, aby bylo možno vytvořit implementaci funkčních požadavků, které se nacházejí v dekompozici funkční analýzy o úroveň níže. Takovýto rozpis implementačních úkonů každého funkčního požadavku vedl k tomu, že bylo možno cíleně a efektivně postupovat při vytváření zdrojového kódu aplikace.

Dekompozice analýzy funkčních požadavků uvedené v předchozím obrázku (Obr. 28) není kompletní, avšak kvůli zachování čitelnosti nebyla rozvedena do nižších úrovní. Pro demonstraci dekompozice funkční analýzy do další úrovně je vybrán funkční požadavek „*Zpracování příkazů od uživatele*“ (Obr. 29). Ten opět obsahuje implementační úkony potřebné k realizaci, což je zejména návrh konkrétní tabulky v databázi, pomocí které lze mezi uživatelem (webovou aplikací) a řídicí jednotkou komunikovat. Zmíněný návrh lze nalézt v následující podkapitole. K povšimnutí stojí, že funkční požadavky nejnižší úrovně již do značné míry korespondují s konkrétními funkcemi (myšleno funkcemi jako podprogramy pro zdrojový kód), které bylo potřeba implementovat.

Obr. 29 - Dekompozice funkčního požadavku na další úroveň

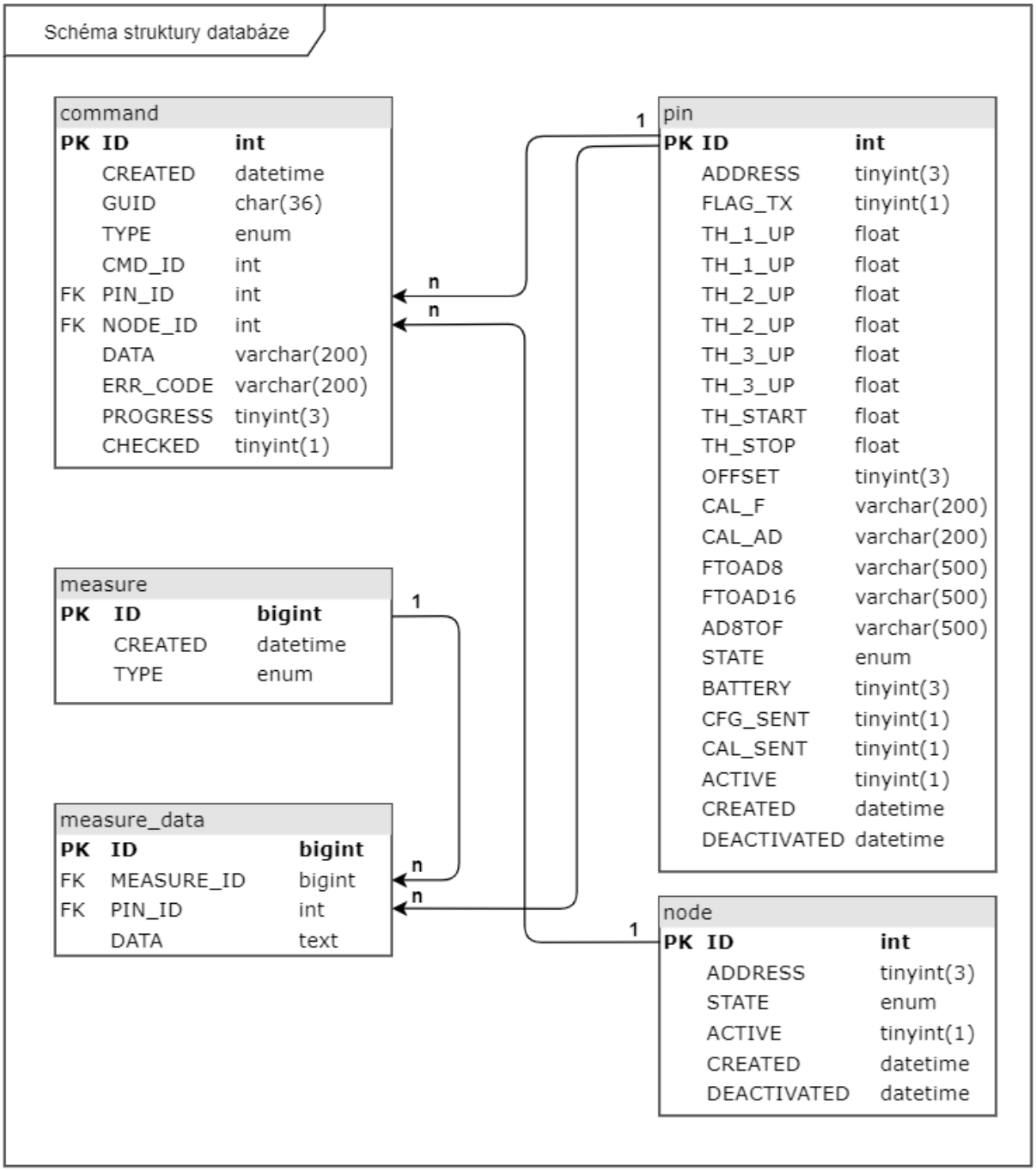
7.2 Komunikace s uživatelem prostřednictvím databáze

Rozhraním pro komunikaci s řídicí aplikací je relační databáze MySQL. Rád bych v tomto bodě odkázal na diplomovou práci Ing. Radima Kuly, která se zabývá tvorbou uživatelského rozhraní a návrhem struktury databáze pro celý měřicí systém [17]. Jelikož se jeho práce primárně zabývala komunikací mezi uživatelským rozhraním a databází, nikoli však mezi databází a řídicí aplikací, bylo zapotřebí provést několik změn. Jednalo se však pouze o změny atributů nebo přidávání nových atributů v jednotlivých tabulkách, nikoliv o úpravy struktury celé databáze. Všechny tyto změny jsou promítnuty do ER diagramu (Obr. 30), který zobrazuje finální verzi struktury databáze.

Ve struktuře databáze se nachází celkem 5 tabulek. Vzhledem k nutnosti odkazovat v některých tabulkách na jiné tabulky vznikají mezi tabulkami relace. Ve struktuře se nachází pouze relace typu 1:*n*,

což znamená, že na jeden záznam v dané tabulce může odkazovat *n* záznamů z jiné tabulky. Pojmenování a popis funkce těchto tabulek je následující:

- command - tabulka pro příkazy pro měřicí systém.
- measure - tabulka pro ukládání měření.
- measure_data - tabulka pro ukládání naměřených dat.
- pin - tabulka měřicích kolíků.
- node - tabulka IR nodů.



Obr. 30 - ER diagram databáze

Nerad bych zde popisoval veškeré změny, které příliš neovlivňují návrh řídicí aplikace nebo nesouvisí s jeho tvorbou, proto zde bude zmínka zejména o tabulce s názvem **command**. Ta slouží pro interakci s uživateli a také podstoupila nejvíce změn. V následující tabulce (Tabulka 6) je uveden popis jednotlivých atributů tabulky command.

Tabulka 6: Datový slovník tabulky command

Atribut	Datový typ	Null	Výchozí hodnota	Popis
ID	int	N		Primární klíč
CREATED	datetime	N	current time	Datum a čas vytvoření příkazu
GUID	char	N		Unikátní identifikátor příkazu
TYPE	enum	N		Typ příkazu
CMD_ID	int	N		Identifikátor příkazu
PIN_ID	int	A	NULL	Cizí klíč (pin → ID)
NODE_ID	int	A	NULL	Cizí klíč (node → ID)
DATA	varchar	N	"	Datové pole příkazu
ERR_CODE	varchar	A	NULL	Chybový kód
PROGRESS	tinyint	A	100	Stav vykonávání příkazu
CHECKED	tinyint	N	0	Zda druhá strana zaznamenala příkaz

Každá akce, kterou požaduje uživatel provést, je doprovázena vygenerováním nového záznamu do tabulky command. Typ (*TYPE*) tohoto záznamu musí být zvolen jako požadavek (request). Příkaz je identifikován pomocí hodnoty ve sloupci *CMD_ID*.

Příkaz typu request s sebou nese také unikátní identifikátor (*GUID*), podle něhož lze jednoznačně identifikovat odpověď (response). Ta totiž nemusí být vložena jako další položka do tabulky command, proto je párování požadavku a odpovědi realizováno pomocí *GUID*u.

Příkaz dále obsahuje reference (cizí klíč) na měřicí kolík a IR node (*PIN_ID, NODE_ID*) do příslušné tabulky. Pomocí této reference lze jednoznačně určit, se kterým zařízením se v rámci příkazu má komunikovat. Hodnoty těchto polí nesmí být vyplněny současně (v žádném scénáři nedochází ke komunikaci s adresovaným měřicím kolíkem i IR nodem), avšak mohou být oba atributy nedefinovány, tedy hodnota těchto polí bude NULL.

Je-li pro vykonání příkazu potřeba zadat parametry, jsou tyto hodnoty zadány do pole *DATA*. Tento atribut je v databázi definován jako datový typ varchar s omezením pro 200 znaků, parametry se zde zadávají ve formátu [*parametr1, parametr2, …, parametrN*]. Přetypování parametrů v řídicí aplikaci je na typ int32, hodnoty tedy musí dodržet rozsah tohoto datového typu.

Pole *CHECKED* vloženého požadavku musí mít vždy hodnotu FALSE (0). Důvodem je, že řídicí aplikace podle hodnoty tohoto pole rozeznává nové příkazy. Jakmile je požadavek příkazu řídicí aplikací zaznamenán, je hodnota tohoto pole nastavena jako TRUE (1).

Pole *ERR_CODE* a *PROGRESS* se u požadavků příkazů nepoužívají, neboť pro něj nemají žádný význam. Pro přehlednost a zachování konzistence dat je vhodné, aby tato pole byla nastavena na hodnotu NULL.

Na každý požadovaný příkaz musí řídicí aplikace nějakým způsobem reagovat. To znamená, že je-li vložen požadovaný příkaz do tabulky command, vždy musí být řídicí aplikací vložena do tabulky odpověď (response) na tento požadavek. Pole *GUID, CMD_ID, PIN_ID* a *NODE_ID* odpovědi jsou vyplněna stejně, jako byla vyplněna u požadavku (requestu). Vrací-li příkaz nějaká data, pak jsou vložena do pole *DATA* ve formátu, který byl uveden v jednom z předchozích odstavců.

Pokud nebylo možno z nějakých důvodů příkaz provést nebo pakliže nastane při provádění neočekávaná chyba, je tato informace zaznamenána do pole *ERR_CODE*. Jako nejjednodušší příklad neúspěšně vykonaného příkazu lze uvést pokus vykonat akci s hodnotou *CMD_ID*, která není implementována.

Pole *PROGRESS* slouží pro jednoduchou procentuální indikaci stavu, ve kterém se provádění příkazu nachází. Majoritní většina příkazů je implementována tak, že při vložení odpovědi je rovnou hodnota pole PROGRESS nastavena na 100, což odpovídá úplnému dokončení příkazu. Toto pole však nachází využití u měřicích módů, kdy po proběhnutí měření dochází k vyčítání naměřených dat z měřicích kolíků, což může v případech dlouhého měření trvat i několik desítek vteřin. V tomto případě je hodnota pole aktualizována při průběžném získávání dat.

Pro ilustraci popsaného principu funkce tabulky příkazů je níže vložen obrázek (Obr. 31), který byl pořízen z nástroje phpMyAdmin, což je populární nástroj umožňující jednoduchou správu obsahu MySQL databáze přes webové rozhraní. Na obrázku lze vidět v tabulce 4 záznamy, přičemž 2 z nich jsou vložené požadavky na příkaz (*ID* = 161 a 163) a 2 jsou příslušné odpovědi k těmto požadavkům (*ID* = 162 a 164). První z příkazů (*ID* = 161 a 162) obsahoval ve svém požadavku parametry a zároveň byly v odpovědi vrácena data, v obou případech jsou tyto informace uvedeny v atributu *DATA*.

ID	CREATED moment of sending command	GUID unique identifier of command	TYPE request/response	CMD_ID Command ID	PIN_ID Pin ID	NODE_ID Node ID	DATA Data	ERR_CODE Response error code	PROGRESS Command complete progress	CHECKED Checked by rPi
161	2019-11-12 14:03:46	4cd1ffe9-d4cc-4a0b-a9b4-57cd1a04f005	request	5	*NULL*	*NULL*	[10,1200]	*NULL*	*NULL*	1
162	2019-11-12 14:03:51	4cd1ffe9-d4cc-4a0b-a9b4-57cd1a04f005	response	5	*NULL*	*NULL*	[32,33]	0 - Successful	100	1
163	2019-11-12 14:05:12	53958e68-b787-4503-b5f4-5e283384588a	request	1	*NULL*	*NULL*		*NULL*	*NULL*	1
164	2019-11-12 14:05:12	53958e68-b787-4503-b5f4-5e283384588a	response	1	*NULL*	*NULL*		0 - Successful	100	1

Obr. 31 - Příklad z tabulky příkazů (command) v nástroji phpMyAdmin

Kompletní výčet všech příkazů pro měřicí systém rozlišovaných dle atributu *CMD_ID* je uveden v příloze této práce. Zároveň je v příloze uveden také výčet možných chyb při zpracování těchto příkazů, které jsou do tabulky vkládány u odpovědi do atributu *ERR_CODE*.

7.3 Externí prostředky

Na základě analýzy funkčních požadavků, která je provedena v úvodu kapitoly 7, bylo zapotřebí jako jeden z prvních kroků zprovoznit komunikaci s databází a také s periférií SPI, přičemž vše muselo být kompatibilní se zvoleným programovacím jazykem C++. Jelikož se jedná o typické technologické prostředky, byla provedena rešerše dostupných řešení a externích knihoven na internetu.

Vhodným prostředkem pro programovací jazyk C/C++ pro přístup k MySQL databázi byl zvolen libmysqlclient, což je jedna z implementací MySQL C API [18]. Prostředek lze do zařízení získat instalací programu MySQL Server nebo programu Connector/C (klientská verze). Jelikož je databáze umístěna přímo v řídicí jednotce, byl vybrán MySQL Server. Druhá zmíněná možnost by byl vhodná, pokud by se databáze nacházela jinde v síti.

Po úspěšné instalaci pro využití funkcí z výše zmíněné knihovny bylo potřeba v kódu pouze připojit hlavičkový soubor *mysql/mysql.h*. Zde jsou obsaženy všechny obvyklé funkce pro práci s databází, jako je provedení připojení, čtení, zápis, mazání, úprava záznamů, provádění transakcí a mnoho dalších funkcí.

Co se týče bezpečnosti knihovny při využití jejich funkcí z více vláken, je na oficiálních stránkách uvedeno, že knihovna je bezpečná pro každé databázové připojení. Pokud by bylo nutné sdílet jedno připojení mezi více vlákny, je zapotřebí ošetřit některé její funkce synchronizačním prostředkem, konkrétně je doporučeno využití mutexu.

Zprovoznění periférie SPI skýtalo několik kroků, jelikož HW řešení bylo vytvořeno pro kanál SPI 1 (též Auxilary SPI), který se nachází na GPIO pinech 35 (MISO), 38 (MOSI) a 40 (SCLK) (dle označení mikrokontroleru Broadcom odpovídají tyto GPIO piny pinům BCM19, 20, 21). Jedná se o alternativní kanál SPI, což znamená, že piny jsou ve výchozím stavu nastaveny pro jinou činnost.

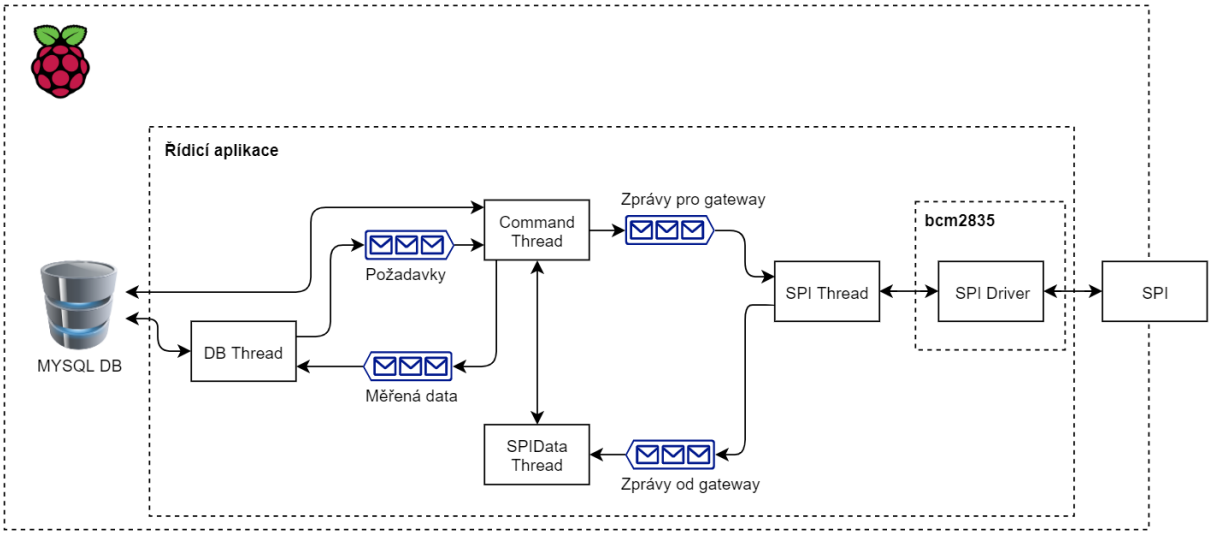
Pro zprovoznění SPI periferie na řídicí jednotce bylo nutné nejprve periferii povolit. Samotné povolení však umožní přístup pouze pro kanál SPI 0, proto bylo potřeba manuálně povolit také druhý kanál (SPI 1). Tento pomocný kanál má k dispozici 3 piny pro Chip Select (volbu slave zařízení), bylo tedy vhodné povolit pouze pin, který je v práci využíván, což je GPIO pin 36 (BCM16). Toto nastavení bylo provedeno úpravou konfiguračního souboru ve složce boot (absolutní cesta k souboru - */boot/config.txt*). Do tohoto souboru byly přidány následující řádky:

```
dtparam=spi=on
dtoverlay=spi1-1cs,cs0_pin=16
```

Dalším krokem bylo vyhledat vhodnou knihovnu pro obsluhu SPI periférie. Jelikož se jedná o alternativní kanál SPI, mnoho volně dostupných knihoven jej neřeší. Nakonec byla pro obsluhu SPI periférie v aplikaci vybrána volně dostupná knihovna s názvem bcm2835, která je zároveň kompatibilní s programovacím jazykem C++ [19]. Knihovnu bylo nejprve potřeba nainstalovat, aby bylo možno využívat její funkce. Po provedení instalace této knihovny je pro využití funkcí potřeba ve zdrojovém kódu připojit hlavičkový soubor *bcm2835.h*. Použitou verzí knihovny při řešení práce byla nejaktuálnější verze 1.58.

7.4 Návrh a implementace řídicí aplikace

Struktura řídicí aplikace byla na základě analýzy funkčních požadavků navržena více vláknově. Vlákna jsou podporována samotným standardem jazyka C/C++, avšak vzhledem k tomu, že řídicí aplikace poběží pod Unix-like operačním systémem, byl zvolen standard POSIX Threads (pthreads), jehož implementaci gcc kompilátor umožňuje. Blokový diagram struktury řídicí aplikace na základě jejich vláken je uveden níže (Obr. 32). Koncept návrhu vychází z modelu Peer, kdy vlákna běží v nekonečné smyčce bez specifického vedoucího (první vlákno se stává rovnocenným vláknem ostatním) a každé vlákno je zodpovědné za své výstupy, které mohou být také vstupy pro jiné vlákno. Dle tohoto modelu byla aplikace rozdělena na 4 vlákna, kde každému z nich byla přiřazena nezávislá úloha. Vlákna jsou dle své úlohy pojmenována, konkrétní popis činností je uveden v podkapitole 7.4.1.



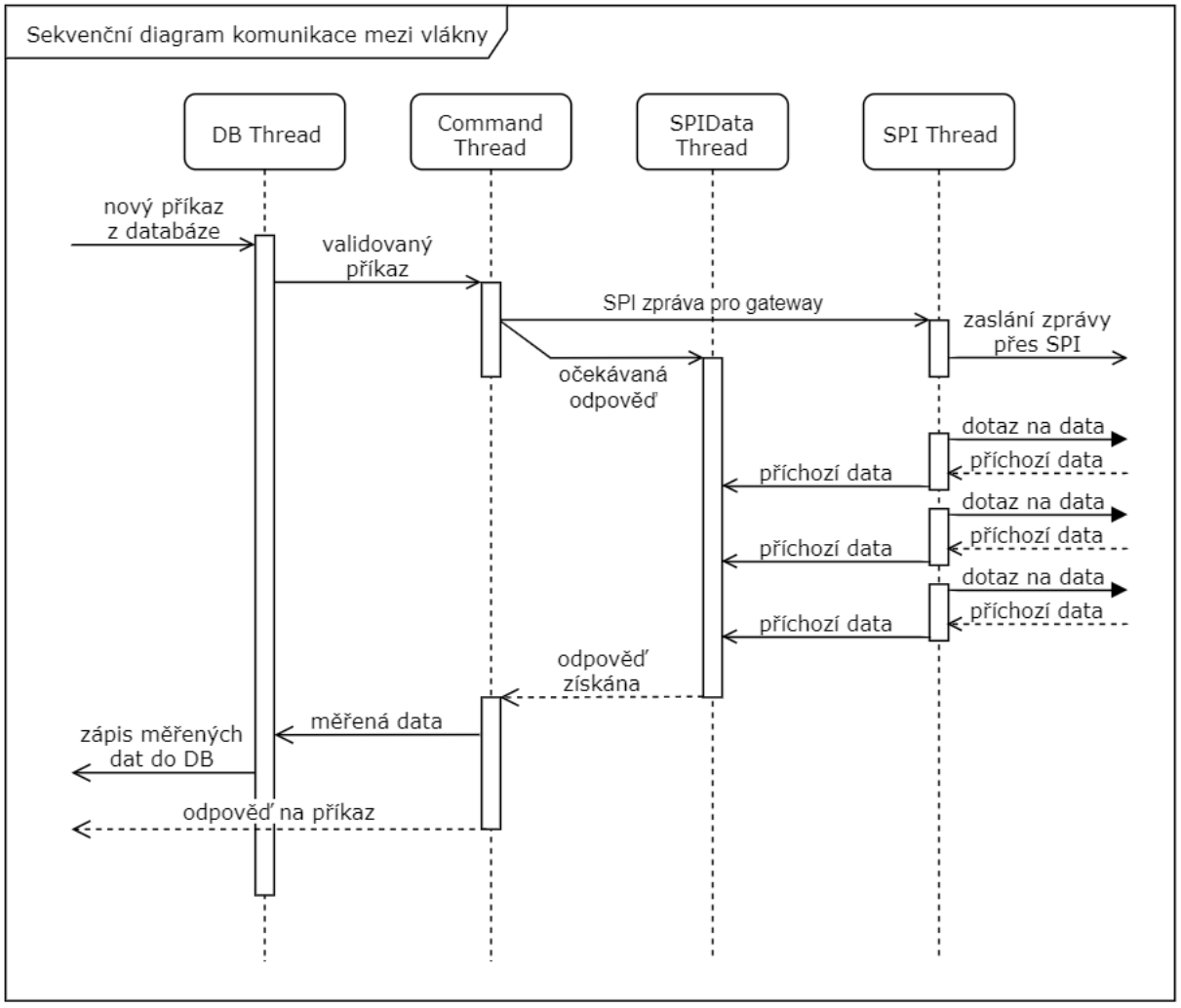
Obr. 32 - Blokové schéma struktury řídicí aplikace

Komunikace mezi vlákny je v aplikaci zajištěna více způsoby. Prvním způsobem je sdílení prostředků, kdy přístup k nim je chráněn synchronizačními mechanismy. Nejvyužívanějším synchronizačním prostředkem v aplikaci je mutex (tj. vzájemné vyloučení). Jedná se o mechanismus, který zabraňuje tomu, aby sdílený prostředek využívalo ve stejnou dobu více než jedno vlákno. Nejpočetnějším typem sdílených prostředků v aplikaci jsou fronty. Pro tento účel byla implementována generická fronta, tedy šablona třídy, kde funkce pro práci s frontou (např. odebírání a přidávání prvků) jsou zamykány pomocí mutexu. Tím lze kdekoli v kódu programu jednoduše vytvořit bezpečnou frontu jakéhokoliv datového typu. Dalším typem sdílených prostředků jsou struktury, které jsou také vnitřně vybaveny mutexem. Pro jakýkoliv zápis nebo čtení dat dané instance struktury je potřeba nejprve uzamknout její mutex, čímž je vyloučen možný souběh činností nad těmito daty.

Druhým způsobem komunikace mezi vlákny v aplikaci je využití podmínkové proměnné. Tu lze využít jedině společně s mutexem, přičemž její využití nespočívá k synchronizaci přístupu ke sdíleným prostředkům více vlákny, ale k synchronizaci vykonávání činností mezi vlákny. Jakmile vlákno při vykonávání programu narazí na podmínkovou proměnnou, suspenduje provádění dalších kroků a čeká, až dostane signál od jiného vlákna, že může v činnosti pokračovat. Standard POSIX threads však umožňuje ještě sofistikovanější způsob užití podmínkové proměnné, který je v řídicí aplikaci využit.

Tím způsobem je využití podmínkové proměnné s časovým limitem. To přináší možnost suspendovat činnost vlákna maximálně na dobu předem stanovenou. Tedy jinými slovy, pokračování činnosti vlákna nastane buď v případě, že suspendované vlákno dostane signál od jiného vlákna, nebo v případě, že stanovená maximální doba suspendace vyprší.

Koncepce rozdělení řídcí aplikace na více vláken lze velmi dobře demonstrovat na příkladu pomocí UML sekvenčního diagramu (Obr. 33). Životní cyklus požadavku v řídicí aplikaci začíná jeho vyčtením z databáze. Následuje jeho zpracování, což je sekvence postupných kroků, mezi které patří například validace požadavku, získání dat z databáze, komunikace se zařízením v měřicím řetězci atd.. Příklad znázorňuje situaci, kdy krokem při vykonávání příkazu je právě komunikace s vybraným zařízením v měřicím řetězci a je očekávána odpověď. V po sobě jdoucích krocích vlákna Command Thread je vytvořena struktura očekávané odpovědi a konkrétní zpráva pro dané zařízení. Tato data jsou předána příslušným vláknům, přičemž aktuální vlákno (Command Thread) je suspendováno s využitím podmínkové proměnné s časovým limitem. Jakmile dorazí odpověď od zařízení v měřicím řetězci nebo vyprší časový limit, pokračuje zpracování příkazu a jsou-li provedeny všechny jeho kroky, případně nastala-li v některém kroku chyba, je provádění ukončeno a výsledek zapsán do databáze.



Obr. 33 - UML sekvenční diagram komunikace mezi vlákny řídicí aplikace

7.4.1 Úlohy jednotlivých vláken aplikace

Pracovní pojmenování vláken vychází z úlohy, kterou mají za úkol vykonávat. Názvy vláken jsou tedy následující:

- DB Thread.
- Command Thread (main).
- SPI Thread.
- SPIData Thread.

Prvním ze zmíněných vláken je vlákno s názvem **DB Thread** starající se o periodickou interakci s databází. Hlavním úkolem tohoto vlákna je hlídat, zda do databáze nepřibyl požadavek na provedení příkazu. Tato funkcionalita musí být zajištěna v jakémkoliv stavu aplikace. Důležitost lze demonstrovat například je-li vykonáván příkaz pro měření, přičemž do databáze je zapsán příkaz pro jeho zastavení, na což musí aplikace okamžitě reagovat. Funkcionalita je dosažena periodickým zasíláním SQL dotazu ve tvaru:

```
SELECT COUNT(CHECKED) FROM command WHERE TYPE = 'request' AND
CHECKED = FALSE AND CREATED > DATE_SUB(NOW(), INTERVAL 5 SECOND)
```

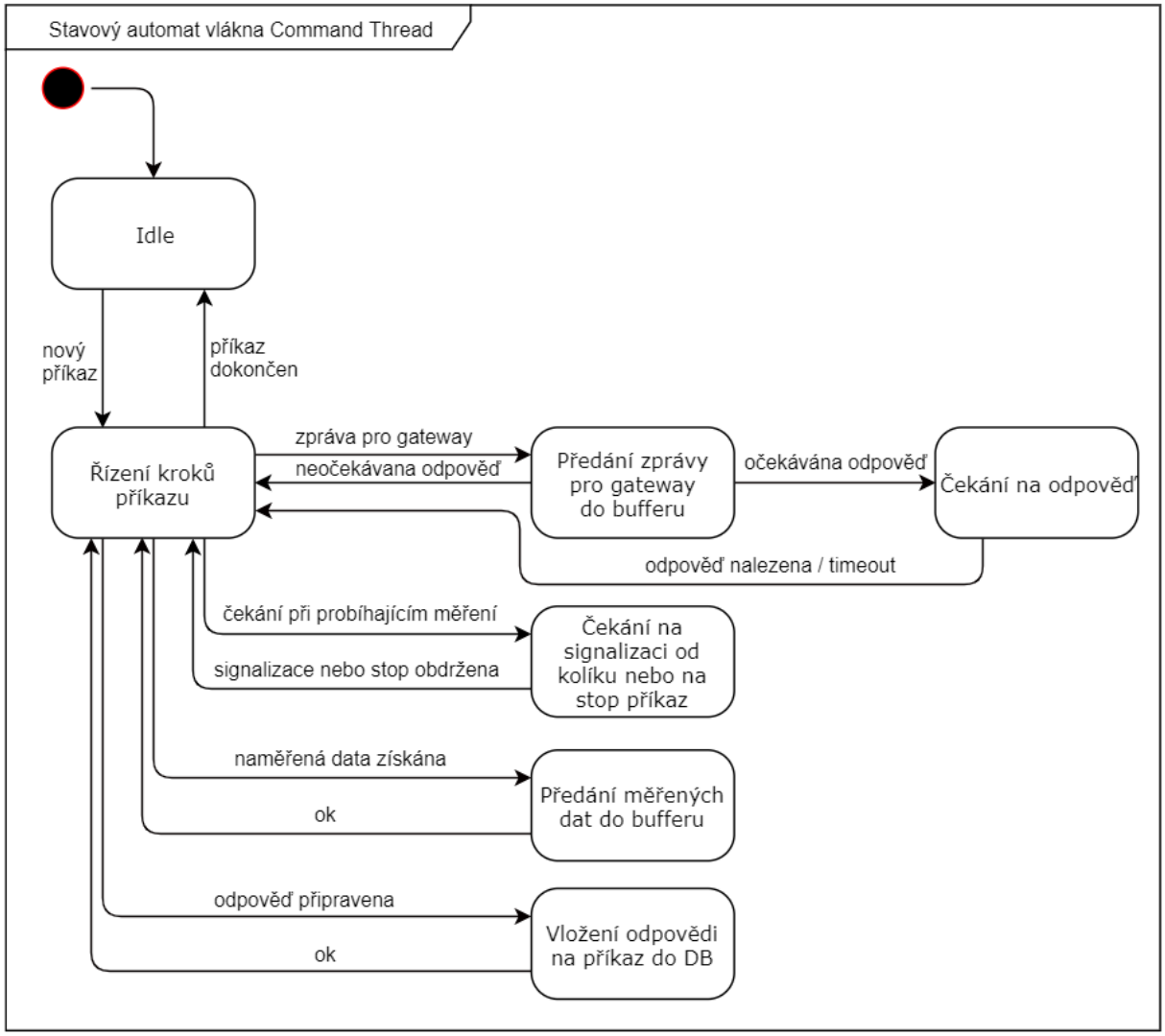
Výstupem tohoto dotazu je počet nových požadavků. Na základě této informace lze dále přistoupit k vyčtení požadavku (od nejstaršího dosud nezpracovaného), přičemž úspěšné vyčtení musí být následováno SQL dotazem, který změní hodnotu ve sloupci *CHECKED* z 0 na 1, jinak by docházelo k opakovanému čtení stejného požadavku.

Vláknu jsou dále přiřazeny i další úlohy nad databází, jako je vkládání naměřených dat či periodická aktualizace atributu *STATE* v tabulce měřicích kolíků a IR nodů (indikace, zda jednotlivá zařízení v měřicím řetězci úspěšně komunikují).

Druhým vláknem v seznamu je **Command Thread**, což je jen jiný název pro hlavní vlákno (main), avšak jelikož nemá dle návrhu žádnou nadřazenou roli vůči ostatním vláknům v aplikaci, je lepší ho pracovně nazývat takto. Jeho rutinou je řídit postup vykonávání jednotlivých validních příkazů, které byly vyčteny z databáze.

Jelikož provádění příkazů může vyžadovat externí data, má i toto vlákno vytvořené své vlastní připojení k databázi, prostřednictvím kterého si aktuálně potřebná data získává, případně data v databázi upravuje. Jako příklad lze uvést potřebu vyčtení konfiguračních dat, vyčtení seznamu aktivních zařízení či aktualizaci adresy zařízení po její úspěšné fyzické změně.

Struktura tohoto vlákna je koncipována jako stavový automat, což je znázorněno na obrázku Obr. 34. Důvodem je, že v průběhu provádění konkrétního příkazu se vlákno dostává do situací, kdy je potřeba provést některou z forem komunikace s jiným vláknem či zapsat data do databáze. Tyto situace při vykonávání příkazu lze označit jako stavy a tím pádem lze využít návrh ve formě stavového automatu.



Obr. 34 - Stavový automat vlákna Command Thread

Dalším vláknem aplikace je **SPI Thread**, jehož úlohou je komunikovat se zařízením gateway. Aby byl zachován bezpečný přístup k SPI sběrnici neboli aby aplikace nepřistupovala k přenosu dat v jeden okamžik z více míst, naskytují se dvě možná řešení. Prvním řešením by bylo veškerý kód, který se stará o přenos dat po SPI ošetřit synchronizačním prostředkem. Druhým řešením je ponechat přístup ke sběrnici pouze jednomu vláknu. Právě druhý způsob řešení byl v aplikaci zvolen.

Práce vlákna spočívá v periodickém dotazování zařízení gateway, zda má ve frontě nějaká data a pokud ano, dojde k jejich přenosu. Zároveň toto vlákno zasílá zprávy, které mají být k zařízení gateway doručeny.

Posledním vláknem řídicí aplikace je **SPIData Thread**. Jeho práce spočívá v procházení příchozích datových rámců od zařízení gateway. Může se nacházet ve dvou stavech, přičemž tento stav je určován vláknem Command Thread v závislosti na tom, zda je očekávána příslušná odpověď od zařízení v měřicím řetězci či nikoliv.

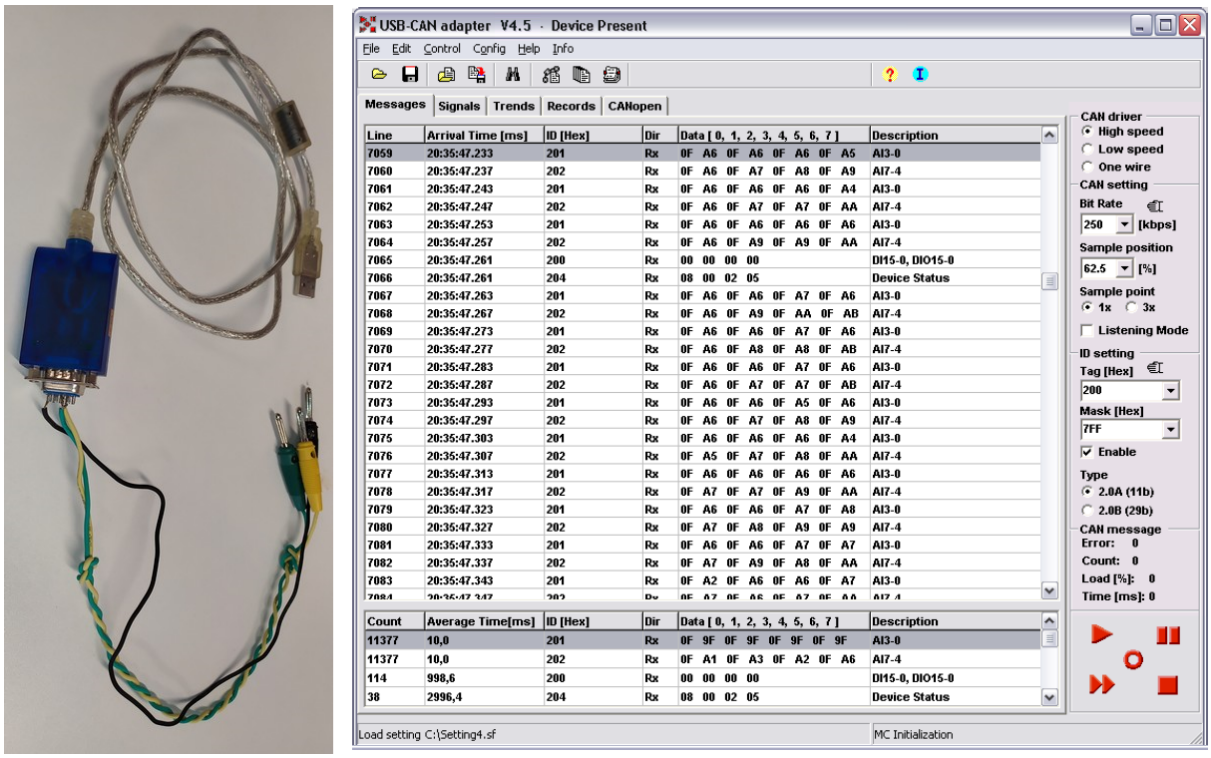
V případě, že odpověď je očekávána, jsou do sdílené struktury pro SPIData Thread předány parametry, jaké má v příchozích datových rámcích hledat a zároveň je změněn jeho stav. Hledanými parametry může být adresa měřicího kolíku, adresa IR nodu, ID příkazu, datová délka zprávy, formát dat (8, 16, 32 bit) či informace, zda je odpověď zasílána v blocích dat. Nastavení těchto parametrů závisí zcela na typu požadavku.

Není-li očekávána žádná odezva od měřicího řetězce, neměly by ze zařízení gateway přicházet žádné jiné zprávy než periodické zprávy od IR nodů znamenající, že jsou aktivní. Na základě těchto zpráv je aktualizována vnitřní tabulka aktivních IR nodů a tím je získán aktuální přehled o stavu těchto zařízení, i pokud neprobíhala delší dobu žádná komunikace.

8 Ladění a testování měřicího systému

Ve fázi vývoje softwaru jednotlivých zařízení bylo zapotřebí odladit jejich chování separátně. U IR nodu byla potřeba odladit funkcionalitu příjmu dat skrze IR přenos. Tato data putují z přijímače na rozhraní UART mikrokontroleru, tudíž jako vhodný ladící prostředek postačil USB/UART FTDI převodník, který je možno připojit na jedné straně do PC pomocí USB a na druhé straně k periferii UART. Pomocí sériového portu lze zasílat data z počítače na rozhraní UART a také opačně, čímž bylo chování IR nodu odladěno.

Pro odladění chování IR nodu a zařízení gateway na sběrnici CAN byl využit USB-CAN adapter – Triple drivers V4.2, který lze připojit k PC pomocí USB a druhým koncem ho lze připojit na sběrnici CAN. Výrobce převodníku zároveň poskytuje vlastní aplikaci pro PC, kterou ho lze jednoduše ovládat. Aplikace umožňuje monitorovat zprávy na sběrnici CAN a zároveň lze zprávy pomocí převodníku zasílat, dokonce lze nastavit i periodu opakování při zasílání zpráv. Této funkcionality bylo využito zejména pro testování zařízení gateway a to tak, že pomocí převodníku byly vysílány zprávy s co nejnižší periodou opakování (minimální nastavitelná hodnota je 1 ms), aby došlo k velkému zatížení sběrnice a bylo pozorováno, zda zařízení gateway je schopna všechny zprávy předat řídicí jednotce. Pomocí tohoto testu byl odladěn jak firmware zařízení gateway, tak řízení komunikace přes sběrnici SPI řídicí jednotkou, kdy výsledkem bylo 100% předání zpráv ze sběrnice CAN do řídicí jednotky.



Obr. 35 – Převodník USB-CAN a aplikace pro jeho ovládání

8.1 Testování přenosu dat v měřicím systému

Po odladění firmwaru IR nodu, zařízení gateway a aplikace řídicí jednotky byl vytvořen řetězec čítající měřicí kolík, 4 IR nody, zařízení gateway a řídicí jednotku. Testován byl přenos dat od řídicí jednotky k měřicímu kolíku a naopak. U řídicí aplikace bylo žádoucí, aby příchozí a odchozí data logovala, čímž bylo možno analyzovat případné chyby v komunikaci. Zároveň byla data vypisována do konzole, aby bylo možno sledovat jejich toky.

Test byl koncipován tak, aby byl z řídicí jednotky zaslán některý z příkazů pro měřicí kolík a byla obdržena patřičná odpověď. Celý tento cyklus započne komunikací řídicí jednotky se zařízením gateway přes sběrnici SPI, následně zařízení gateway pomocí sběrnice CAN vyšle příkaz pro odpojení IR nodů ze sběrnice, zašle IR data směrem k měřicímu kolíku, ten data zpracuje a požadovaný příkaz vykoná. Kolík zašle zpět odpověď pomocí IR, kterou zachytí IR nody, zašlou data skrze sběrnici CAN do zařízení gateway a ta pomocí sběrnice SPI předá data řídicí jednotce.

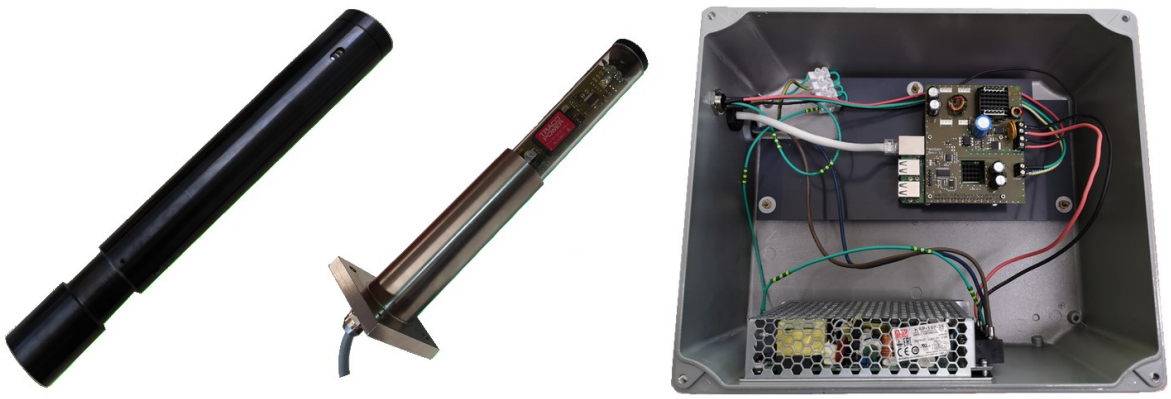
Test byl opakován pro několik příkazů a vždy dopadl úspěšně. Bylo umožněno komunikovat s jakýmkoli IR nodem a měřicím kolíkem v systému. Na následujícím obrázku je zachycen výpis do konzole z programu běžícího v řídicí jednotce, na kterém jsou vidět přijatá data pocházející od měřicího kolíku zachycené čtyřmi IR nody.

```
DEBUG: 33 14     14 14 14 14 13 14 13 14      <- DATA [pin #0, node #29]
DEBUG: 33 14     14 14 14 14 13 14 13 14      <- DATA [pin #0, node #30]
DEBUG: 33 15     14 13 14 13 13 14 14 14      <- DATA [pin #0, node #29]
DEBUG: 33 15     14 13 14 13 13 14 14 14      <- DATA [pin #0, node #30]
DEBUG: 33 16     14 14 0 0 0 0 0 0      <- DATA [pin #0, node #29]
DEBUG: 33 16     14 14 0 0 0 0 0 0      <- DATA [pin #0, node #30]
DEBUG: 33 0    7 10 14 13 14 14 14 14      <- DATA [pin #0, node #12]
DEBUG: 33 0    7 10 14 13 14 14 14 14      <- DATA [pin #0, node #15]
DEBUG: 33 1    14 14 13 14 13 13 14 14      <- DATA [pin #0, node #12]
DEBUG: 33 1    14 14 13 14 13 13 14 14      <- DATA [pin #0, node #15]
```

Obr. 36 - Data přicházející do řídicí jednotky

8.2 Testování měřicích kolíků

Pro testování funkčnosti měřicích kolíků byla vytvořena sestava měřicího systému, která se skládala z finálních prototypů zařízení. Konkrétně byl měřicí řetězec složen ze 4 měřicích kolíků, 6 IR nodů, zařízení gateway a řídicí jednotky.



Obr. 37 – Finální prototypy všech zařízení (vlevo měřicí kolík, uprostřed IR node, vpravo gateway + řídicí jednotka a napájecí zdroj)

Pro možnost zatížení měřicích kolíků byl zapůjčen hydraulický lis s ruční pumpou. Lis dokáže vyvinout zatížení až 12 tun (120 kN), což je pro měřicí kolík dostačující, neboť je koncipován pro maximální zatížení 80 kN. Lis je možno upravit do 8 pracovních úrovní v rozmezí 0-650 mm, což vyhovuje délce měřicího kolíku. Na níže uvedeném obrázku je popisovaný hydraulický lis zachycen i s jeho rozměry.

Obr. 38 - Hydraulický lis pro testování funkčnosti měřicích kolíků

Nevýhodou pro testování bylo, že píst hydraulické pumpy má malý průměr a zejména z bezpečnostních důvodů nebylo možné bez mechanických úprav umístit pod lis více měřicích kolíků najednou. Pro proces testování se však nejednalo o kritickou překážkou, neboť měření všemi měřicími kolíky najednou bylo možno provést, přičemž bylo jasné, že nezatíženými kolíky jsou naměřeny hodnoty odpovídající nulovému zatížení.

Pro testování měřicího systému bylo využito také uživatelské rozhraní, kterým je webová aplikace. Toto rozhraní nebylo vytvořeno v rámci této diplomové práce, nýbrž Ing. Radimem Kulou při tvorbě jeho diplomové práce a následně dále vyvíjeno na základě aktuálních potřeb v rámci průběhu prací na projektu.

Prvním krokem, který bylo potřeba pro měření pomocí kolíku provést, byla jeho kalibrace. Proces kalibrace měřicího kolíku znamená vytvoření matematické funkce pro přepočet měřených hodnot ze snímače na působící zatížení. Pro tento proces bylo pod měřicí kolík umístěno tenzometrické měřidlo, které sloužilo jako reference pro určení aktuálního působícího zatížení vyvinutého hydraulickým lisem. Kalibrační měření na měřicím kolíku zasílá surové hodnoty ze snímače digitalizované pomocí 16bitového AD převodníku. Jako odpovídající kalibrační dvojice hodnot je tedy považována aktuálně zasílána 16bitová surová hodnota z měřicího kolíku (respektive průměr z posledních N hodnot) a aktuální měřené zatížení pomocí tenzometrického měřidla.

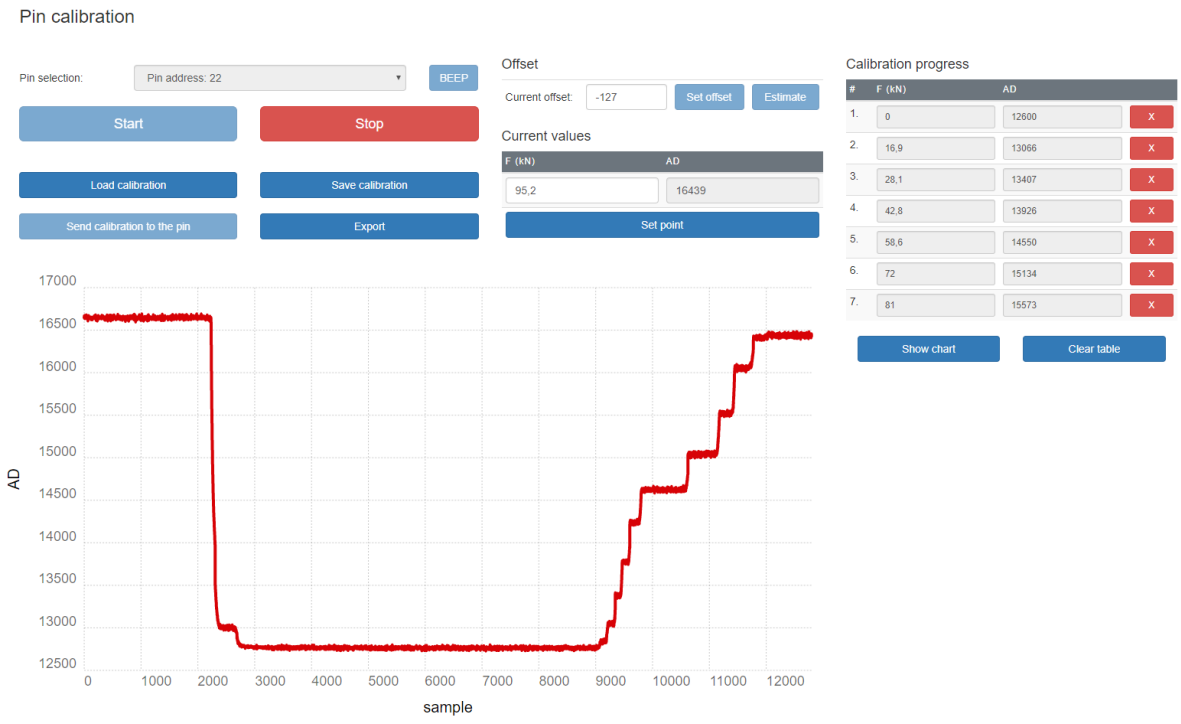
Pro úspěšné provedení kalibrace bylo potřeba zaznamenat alespoň dvě hodnoty ze senzoru pro dvě různá zatížení (neboli vytvořit dvě kalibrační dvojice). V takovém případě by byla kalibrační křivka lineární. Lepším řešením bylo zvolit měření více bodů, aby byly eliminovány nepřesnosti

jednotlivých měření a kalibrační funkce byla vytvořena polynomiální regresí, která užívá pro minimalizaci chyby metodu nejmenších čtverců.

Postup pro kalibraci měřicího kolíku byl následující:

1. Spuštění kalibračního měření na vybraném měřicím kolíku pomocí uživatelského rozhraní.
2. Manuální vynulování tenzometrického měřidla při nulovém zatížení.
3. Opsání údaje z tenzometrického měřidla do uživatelského rozhraní.
4. Zaznamenání dvojice hodnot (surové hodnoty AD převodníku a působícího zatížení).
5. Zvýšení zatížení pumpováním na hydraulickém lisu.
6. Opakování kroků 3, 4 a 5 pro několik různých zatížení.
7. Uložení kalibrace a ukončení kalibračního měření

Výsledkem postupu je tabulka hodnot, která je základem pro stanovení přepočtu surových hodnot na sílu. Na následujícím obrázku je zachycen snímek z uživatelského rozhraní (Obr. 39), pomocí kterého byla kalibrace provedena. Zmíněná tabulka kalibračních hodnot se nachází v pravé části webové aplikace, graf ukazuje průběh surových hodnot zasílaných měřicím kolíkem během kalibračního měření.



Obr. 39 - Ukázka z webové aplikace při kalibraci měřicího kolíku

Po provedení kalibrace je možno naměřená data zasílaná měřicím kolíkem přepočítat na odpovídající zatížení. Lze tedy na měřicím kolíku spustit jakýkoliv měřicí režim, jako je například měření pomocí thresholdů, časově omezené měření a podobně, a získat průběh zatížení v čase.

Závěr

Na základě spoluúčasti na vědecko-výzkumném projektu, který se zaměřoval na vytvoření prototypu měřicího kolíku s integrovaným měřením upínací síly a bezdrátovým přenosem dat, byla definována tato diplomová práce, jejíž cílem bylo vytvořit komunikační protokoly pro přenos dat mezi zařízeními v navrženém měřicím systému, dále vytvořit softwarovou část pro 3 různá zařízení, a nakonec také sestavit a otestovat prototypový měřicí systém.

Prvním z cílů diplomové práce bylo na základě analýzy koncepce měřicího systému navrhnout komunikační protokoly mezi jednotlivými zařízeními měřicího systému. V práci byly navrženy komunikační protokoly pro každé použité komunikační médium, kterými jsou sběrnice SPI, sběrnice CAN a také IR datový přenos. Byla provedena důkladná analýza způsobu přenosu dat v měřicím systému a z této analýzy byly vyvozeny jednotlivé poznatky pro návrh komunikačních protokolů. Komunikační protokoly respektují adresaci zařízení, využívání příkazů pro rozpoznávání požadovaných akcí a mimo protokol sběrnice CAN využívají kontrolní součet jako ochranu pro korektní přenos dat. Sběrnice CAN využívá jako ochranu CRC, která je součástí standardu.

Nejvýraznějším cílem práce byl kompletní vývoj firmware pro zařízení IR node a gateway a software pro řídicí jednotku. U každého zařízení byla nejprve provedena analýza použitých hardwarových prostředků, od kterých se vývoj softwaru odvíjí. Firmware pro zařízení IR node a gateway byl vytvořen v jazyce C pro mikrokontrolery PIC společnosti Microchip, řídicí software byl vytvořen v jazyce C++ pro operační systém Raspbian. Dále byla provedena analýza funkčních požadavků, ze kterých vzešel návrh architektury daného softwaru. V dalším kroku byly popsány některé implementační detaily, které byly v rámci tvorby zvažovány. Pomocí modelovacích nástrojů pro software bylo vytvořeno několik diagramů (zejména pak v jazyce UML), které zachycují realizovaný systém, čímž slouží jako dokumentace návrhu a usnadňují pochopení systému.

Jedním z cílů bylo také provést testování celého měřicího systému složeného z výsledných prototypů. Došlo tedy ke spojení mechanické, hardwarové a softwarové části řešení projektu a byl sestaven měřicí systém. Testování mělo za cíl zjistit, zda přenos dat mezi zařízeními v měřicím systému probíhá úspěšně a lze měřicí systém ovládat pomocí uživatelského rozhraní. Zároveň byla pomocí ručního hydraulického lisu otestována funkčnost měřicích kolíků. Oba tyto testy dopadly úspěšně. Pomocí uživatelského rozhraní ve webové aplikaci byl ovládán měřicí systém, následně vyvinutá síla hydraulickým lisem byla naměřena měřicím kolíkem a tato data úspěšně doputovala měřicím systémem až do databáze, odkud se zobrazila také ve webové aplikaci.

Dalším budoucím postupem je ověření funkčnosti prototypu měřicího systému v cílovém prostředí. Po zkušenosti s použitím v průmyslovém prostředí mohou vyplynout nové poznatky, které budou mít dopad na některou část měřicího systému, případně se mohou objevit návrhy na změny, které by cílily na zlepšení uživatelského komfortu, použitelnosti, spolehlivosti, bezpečnosti a podobně. Z hlediska softwarové části měřicího systému jsou změny jednoduše aplikovatelné, neboť lze přehrát firmware či software i v již používaných zařízeních.

Zadání diplomové práce se podařilo naplnit ve všech jejích bodech. Průběh tvorby diplomové práce, jakožto i průběh práce v rámci řešení projektu, byl pravidelně konzultován s vedoucím diplomové práce a se členy projektového týmu.

Použitá literatura

[1] KAJZAR, Daniel. *Návrh designu elektroniky pro měření tlaku* [online]. Ostrava, 2019 [cit. 2020-01-15]. Dostupné z: http://hdl.handle.net/10084/136159. Bakalářská práce. Vysoká škola báňská – Technická univerzita Ostrava.

[2] RASPBERRY PI FOUNDATION. *Raspberry Pi 4 Tech Specs* [online]. 2019 [cit. 2020-01-15]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[3] ANTOSZYK, Marián. *Regulace vybrané soustavy založená na technologii IQRF* [online]. Ostrava, 2018 [cit. 2020-01-15]. Dostupné z: http://hdl.handle.net/10084/128617. Bakalářská práce. Vysoká škola báňská – Technická univerzita Ostrava.

[4] ŞAFAK, Mehmet. *Digital communications*. Chichester: Wiley, 2017. 884 stran. ISBN 978-1-119-09125-7.

[5] Vývoj.HW.cz. *Teorie datového IR přenosu* [online]. 1998 [cit. 2020-01-20]. Dostupné z: https://vyvoj.hw.cz/teorie-a-praxe/dokumentace/teorie-datoveho-ir-prenosu.html

[6] Vishay Semiconductors. *IR Receiver Modules for Remote Control Systems*. [online]. 2015 [cit. 2020-01-20]. Dostupné z: http://www.farnell.com/datasheets/2049436.pdf

[7] Texas Instruments. *Infrared Remote Control Modulation and Encoding Theory*. [online]. 2015 [cit. 2020-01-22]. Dostupné z: http://www.ti.com/lit/an/slaa644b/slaa644b.pdf

[8] IBRAHIM, Dogan. *Chapter 9. Advanced PIC18 Projects - CAN Bus Projects*. [online]. 2008 [cit. 2020-01-18]. Dostupné z: https://www.researchgate.net/publication/279453244

[9] ISO 11898: *Road vehicles - Controller are network (CAN).* Switzerland: International Organization for Standardization, 2015 [cit. 2020-01-18].

[10] Wikipedia: The free encyclopedia. *Serial Peripheral Interface* [online]. 2020 [cit. 2020-01-18]. Dostupné z: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

[11] SOMMERVILLE, Ian. *Softwarové inženýrství.* Přeložil Jakub GONER. Brno: Computer Press, 2013. 680 stran. ISBN 978-80-251-3826-7.

[12] Object Management Group. *Unified Modeling Language* [online]. 2017 [cit. 2020-02-01]. Dostupné z: https://www.omg.org/spec/UML/2.5.1/PDF

[13] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.* Přeložil Bogdan KISZKA. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

[14] Smartdraw. *Data Flow Diagram* [online]. [cit. 2020-02-05]. Dostupné z: https://www.smartdraw.com/data-flow-diagram/

[15] ISO/IEC 15909: *Software and system engineering - High-level Petri nets*. Switzerland: International Organization for Standardization, 2004 [cit. 2020-02-05].

[16] Microchip. *Development tools* [online]. 2019 [cit. 2019-12-15]. Dostupné z: https://www.microchip.com/development-tools/

[17] KULA, Radim. *Návrh a implementace front-end pro konfiguraci zařízení měření tlaku* [online]. Ostrava, 2019 [cit. 2020-01-18]. Dostupné z: http://hdl.handle.net/10084/136173. Diplomová práce. Vysoká škola báňská - Technická univerzita Ostrava.

[18] MySQL. *MySQL 8.0 Reference Manual* [online]. 2019 [cit. 2019-12-12]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/

[19] AirSpayce. *bcm2835 1.58* [online]. 2019 [cit. 2019-05-15]. Dostupné z: https://www.airspayce.com/mikem/bcm2835/

Seznam příloh

Součástí DP jsou elektronické přílohy.

Adresářová struktura elektronických příloh:

- A_Popis_prikazu_mericiho_systemu – Popis ovládacích příkazů pro měřicí systém v českém a anglickém jazyce
- B_Struktura_DB – Popis struktury databáze v českém a anglickém jazyce
- C_Obrazky – Obrázky použité v DP a podklady pro vlastní obrázky
- D_Komunikacni_protokoly – Příkazy pro zařízení dle komunikačních protokolů

Příloha A: *Příkazy měřicího systému*

Tabulka rozpoznatelných příkazů pro měřicí systém

ID	Název
1	Kontrola stavu měřicích kolíků
2	Zápis konfigurace a kalibrace do pinu
3	Real-time režim jednoho kolíku, formát dat 8bit
4	Režim spouštění měření časem, analýza
5	Režim spouštění měření časem, analýza + graf
6	Režim spouštění thresholdem, analýza
7	Režim spouštění thresholdem, analýza + graf
8	Změna adresy měřicího kolíku
9	Změna adresy IR nodu
10	Stop měření
11	Zvukové znamení měřicího kolíku
12	Real-time režim jednoho kolíku, formát dat 16bit
13	Zápis kalibrace do měřicího kolíku
14	Odhad offsetu daného měřicího kolíku
15	Zápis offsetu do měřicího kolíku

Příloha B: *Chybové kódy příkazů měřicího systému*

Tabulka možných chybových kódů

Kód	Význam
0	**Příkaz úspěšně vykonán**
1	**Vykonávání příkazu přerušeno uživatelem** *(např. měření, které bylo zastaveno před dokončením příkazem pro zastavení měření*)
2	**Časový limit pro komunikaci s vybraným zařízením vypršel**
3	**Nenalezen žádný aktivní pin** (*prázdná tabulka pinů*)
4	**Nová adresa zařízení je mimo povolené limity**
20	**Špatný požadavek –** Špatný datový vstup (DATA) požadavku *(např. více než 1 hodnota při změně adresy zařízení, hodnota mimo rozsah typu int32…*)
21	**Špatný požadavek –** PIN_ID požadavku nemůže být NULL
22	**Špatný požadavek –** NODE_ID požadavku nemůže být NULL
23	**Špatný požadavek –** NODE_ID a PIN_ID požadavku jsou vyplněny zároveň
24	**Špatný požadavek –** Neznámé CMD_ID požadavku
40	**Špatný požadavek měření –** Požadovaná doba do začátku měření mimo limity
41	**Špatný požadavek měření –** Požadovaná doba měření mimo limity
42	**Špatný požadavek měření –** Časový limit měření mimo limity
60	**Špatná konfigurační data –** V databázi nejsou uložena data kalibrace pinu
61	**Špatná konfigurační data –** Data kalibrace pinu jsou špatná
62	**Špatná konfigurační data –** Thresholdy jsou nastaveny špatně
63	**Špatná konfigurační data –** Offset pinu je mimo limity
80	**Změna záznamu v databázi selhala**
81	**Vložení záznamu do databáze selhalo**
99	**Neočekávaná chyba** *(např. vyčtení ID právě vloženého měření selhalo (to znemožní vložení naměřených dat))*